ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ
ΕΡΓΑΣΙΑ

«Ανάπτυξη Διαδικτυακής Εφαρμογής Διαχείρισης Εισιτηρίων Πολιτιστικών Εκδηλώσεων»

Της φοιτήτριας
Χαλιπήλια Βασιλικής
Αρ. Μητρώου: it103660

Επιβλέπων
Κώστογλου Βασίλης
Καθηγητής

Φλεβάρης 2024

Τίτλος Δ.Ε.: Ανάπτυξη Διαδικτυακής Εφαρμογής Διαχείρισης Κράτησης Εισιτηρίων Πολιτιστικών Εκδηλώσεων
Κωδικός Δ.Ε 23221
Ονοματεπώνυμο φοιτητή: Χαλιπήλια Βασιλική
Ονοματεπώνυμο εισηγητή Κώστογλου Βασίλης
Ημερομηνία ανάληψης Δ.Ε.: 21-5-2023
Ημερομηνία περάτωσης Δ.Ε: 21-2-2024

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Ευχαριστίες

Η ολοκλήρωση της παρούσας διπλωματικής δε θα μπορούσε να έχει υλοποιηθεί χωρίς την υποστήριξη του καθηγητή μου Βασίλη Κώστογλου που θα ήθελα να του εκφράσω ένα μεγάλο ευχαριστώ για την βοήθεια που μου πρόσφερε.

Στη συνέχεια θα ήθελα να ευχαριστήσω τους γονείς μου Σπύρο Μυλωνά & Νούλα Γεωργιάδου που ήταν δίπλα μου κατά την διάρκεια όλων αυτών των χρόνων, προσφέροντάς μου απλόχερα κάθε είδους βοήθεια.

Τέλος, οφείλω ένα μεγάλο ευχαριστώ και μια συγνώμη στα παιδάκια μου Γιώργο Πανίδη & Ειρήνη Πανίδου τα οποία αρκετές φορές παραμέλησα λόγο αρκετών υποχρεώσεων..

Περίληψη

Καθώς στη σημερινή εποχή, η αλληλεπίδραση μας σε όλες τις πτυχές της καθημερινής μας ζωής τείνει να ψηφιοποιηθεί προς όφελος της ευκολίας και άνεσης μας, η κράτηση εισιτηρίων σε πολιτιστικές εκδηλώσεις με τη χρήση της τεχνολογίας μας δίνει επιπλέον δυνατότητες διαχείρισης, ενημέρωσης για τις τρέχουσες και τις επερχόμενες εκδηλώσεις, καθώς και για τη διαθεσιμότητα των εισιτηρίων σε κάθε στιγμή. Η παρούσα διπλωματική εργασία επικεντρώνεται στην ανάπτυξη μιας εφαρμογής διαχείρισης κρατήσεων εισιτηρίων πολιτιστικών εκδηλώσεων πανελλαδικά. Μέσω της χρήσης της στοίβας τεχνολογιών MEAN (MongoDB, Nest.JS/Express.JS, Angular, Node.JS), η εφαρμογή παρέχει μια σύγχρονη και φιλική προς τον χρήστη εμπειρία. Οι χρήστες μπορούν να περιηγηθούν στις εκδηλώσεις, να κάνουν κρατήσεις, και να διαχειριστούν το πρόγραμμα τους με ευκολία, ενώ οι διαχειριστές/διοργανωτές έχουν πρόσβαση σε ένα διαχειριστικό περιβάλλον για την διαχείριση των εκδηλώσεων και των εισιτηρίων τους. Επιπλέον, αναλύονται οι απαιτήσεις της εφαρμογής, περιγράφεται διαγραμματικά η ανάπτυξη της και αξιολογούνται οι στόχοι της ως προς την εκπλήρωση της ανάγκης χρήσης της τεχνολογίας στη βελτίωση της εμπειρίας των θεατών για τη συμμετοχή τους και των διοργανωτών στην αποτελεσματική διαχείριση των εκδηλώσεων.

Λέξεις-Κλειδιά

Διαδικτυακή Εφαρμογή, Διαχείριση κρατήσεων, στοίβα MEAN, MongoDB, Angular, Nest.JS

«Web Application Development for Ticket Reservation Management of Cultural Events»

«Chalipilia Vasiliki»

Abstract

As contemporary daily life tends to be digitized for the benefit of our convenience and comfort, booking tickets to cultural events by utilizing technological means, provides us with additional possibilities to manage, inform about upcoming events, as well about their availability at any time. The current thesis focuses on the development of an application for ticket reservations managing, with regard to cultural events nationwide. Through the use of the MEAN technology stack (MongoDB, Nest.JS, Angular, Node.JS), the application provides a modern and user-friendly experience for its user. Users may browse events, make reservations, and manage their schedule with ease, while administrators/organizers have access to a special interface to manage their events and tickets. In addition, the development process is described, with the requirements analysis, graphical description, and evaluation of the app with regard to the user experience for spectators, as well for admins/organizers.

Key-Words

Web application, Ticket reservation, MEAN stack, MongoDB, Angular, Nest.JS

Περιεχόμενα

Κατάλογος Εικόνων

Κατάλογος Πινάκων

Κατάλογος Ακρονυμίων και Συντμήσεων

Κεφάλαιο 1ο: Περιγραφή του Προβλήματος

1.1 Εισαγωγή

Η σημερινή εποχή έχει διαμορφωθεί από ένα καταιγισμό τεχνολογικών εξελίξεων, επηρεάζοντας τις κοινωνικές και εργασιακές μας δραστηριότητες. Ο κόσμος της ψυχαγωγίας και των πολιτιστικών εκδηλώσεων δεν αποτελεί εξαίρεση, καθώς η εμπειρίες μας στη συμμετοχή τους έχουν αναβαθμιστεί αξιοποιώντας τις τεχνολογικές καινοτομίες προς όφελος της ευκολίας διαχείρισης, προγραμματισμού και ελέγχου τόσο των κρατήσεων από τους θεατές, όσο και οργάνωσης από την πλευρά των διαχειριστών.

Η επικράτηση των διαδικτυακών πλατφορμών στη διαδικασία κράτησης εισιτηρίων έχει μετατρέψει τη διαδικασία σε μια εύκολη και άνετη εμπειρία, επιτρέποντας στους χρήστες να προγραμματίσουν τη συμμετοχή τους με άνεση και αξιοπιστία. Επιπλέον, οι διοργανωτές εκτός από την ικανοποίηση αυτής της τάσης στους θεατές, επωφελούνται από την ευκολία διαχείρισης, ενημέρωσης και οργάνωσης των εκδηλώσεων.

Στο πλαίσιο αυτό, η παρούσα διπλωματική εργασία αποσκοπεί στην ανάπτυξη μιας εφαρμογής διαχείρισης κρατήσεων εισιτηρίων για πολιτιστικές εκδηλώσεις (πχ. θεατρικές παραστάσεις). Οι χρήστες της εφαρμογής θα έχουν τη δυνατότητα να περιηγηθούν στις τρέχουσες και επερχόμενες εκδηλώσεις, και να κάνουν κρατήσεις με απλότητα και ευκολία. Επιπλέον θα παρέχει μια διαχειριστική διεπαφή για τους διοργανωτές/διαχειριστές, όπου θα έχουν τη δυνατότητα να διαχειρίζονται τις πληροφορίες των εκδηλώσεων, να προσθέτουν καινούριες, καθώς και να διαχειρίζονται τα εκάστοτε εισιτήρια τους.

Επιπλέον, στην εργασία παρουσιάζεται η αρχιτεκτονική και οι τεχνολογίες που χρησιμοποιήθηκαν καθώς και η διαδικασία ανάπτυξης της. Τέλος αξιολογείται η εφαρμογή και παρουσιάζονται τα συμπεράσματα της εργασίας.

1.2 Περιγραφή του Προβλήματος

Οι συνεχώς εξελισσόμενες τεχνολογίες στη σημερινή εποχή, έχουν ανατρέψει τον τρόπο με τον οποίο αλληλοεπιδρούμε στην καθημερινή μας ζωή. Ένας τομέας που έχει επηρεαστεί όχι τόσο στο προϊόν όσο στη διαδικασία κρατήσεων εισιτηρίων, είναι και αυτός της ψυχαγωγίας και ιδιαίτερα των πολιτιστικών εκδηλώσεων όπως οι θεατρικές παραστάσεις, Οι παραδοσιακοί τρόποι αγοράς εισιτηρίων και κρατήσεων δεν επαρκούν πλέον για την ικανοποίηση των απαιτήσεων της αγοράς, καθιστώντας επιτακτική την ανάπτυξη νέων και καινοτόμων λύσεων.

Η παρούσα εργασία εστιάζει στον τομέα της ψυχαγωγίας και συγκεκριμένα των πολιτιστικών εκδηλώσεων, αναζητώντας τρόπους βελτίωσης της εμπειρίας κρατήσεων εισιτηρίων. Ειδικότερα, διαπραγματευόμαστε το πρόβλημα της ανάπτυξης μιας εφαρμογής διαχείρισης κρατήσεων για πολιτιστικές εκδηλώσεις.

Η αντιμετώπιση των προκλήσεων που αντιμετωπίζουν οι θεατές καθώς και οι διοργανωτές των παραστάσεων, αποτελεί το επίκεντρο της παρούσας εργασίας. Αφενός για τους θεατές, η επίσκεψη σε φυσικά σημεία πώλησης ή η τηλεφωνική κράτηση, μπορεί να αποδειχθούν χρονοβόροι και μη

πρακτικοί τρόποι για το σύγχρονο κοινό. Αφετέρου, για τους διοργανωτές η διαχείριση των εκδηλώσεων και των κρατήσεων, απαιτεί ικανότητες διαχείρισης πολλών παραμέτρων όπως οι χώροι, οι ημερομηνίες, οι τιμές και οι διαθέσιμες θέσεις, ώστε να διασφαλιστεί η ομαλή διεξαγωγή τους.

Από τα παραπάνω προκύπτει η ανάγκη για την ανάπτυξη μιας εφαρμογής διαχείρισης κρατήσεων, η οποία θα δημιουργεί και ένα διαισθητικό και ευέλικτο σύστημα βελτίωσης της εμπειρίας τόσο των θεατών όσο και των διοργανωτών.

1.3 Συνεισφορά της Διπλωματικής Εργασίας

Η παρούσα εργασία αποσκοπεί στην ανάπτυξη μιας ολοκληρωμένης πλατφόρμας διαχείρισης κρατήσεων για πολιτιστικές εκδηλώσεις, βασισμένη στη στοίβα ανάπτυξης MEAN (MongoDB, Nest.JS/Express.JS, Angular, Node.JS).

Σε έναν ταχύτατα εξελισσόμενο τομέα όπως είναι αυτός της ψυχαγωγίας, η εξυπηρέτηση των αναγκών των θεατών όσο και των διοργανωτών απαιτεί καινοτόμες λύσεις που συνδυάζουν την πρακτικότητα με την βελτιωμένη εμπειρία χρήσης, όπως θα μπορούσε να προσφέρει μια εφαρμογή διαχείρισης κρατήσεων.

Οι ανάγκες αυτές προκύπτουν από την απαίτηση για αμεσότητα, εγκυρότητα, ευκολία διαχείρισης και παρακολούθηση ιστορικότητας. Οι θεατές δεν απαιτείται να περιμένουν σε ουρές αναμονής ή στο ακουστικό για την κράτηση ή ακύρωση ενός εισιτηρίου καθώς για την ενημέρωση των τρεχουσών εκδηλώσεων και των διαθέσιμων θέσεων. Επιπλέον οι διοργανωτές δεν είναι απαραίτητο πλέον να διακατέχονται από εξεζητημένες ικανότητες απομνημόνευσης και διαχείρισης της οργάνωσης των εκδηλώσεων, παρά μόνο εξοικείωση με τη φιλικό προς το χρήστη περιβάλλον διαχείρισης που παρέχεται από την πλατφόρμα.

Με τον λεπτομερή σχεδιασμό, αξιοποιώντας τη γλώσσα μοντελοποίησης UML (Unified Modeling Language) και συγκεκριμένα τα διαγράμματα Περιπτώσεων Χρήσης (Use-case diagram), τα διαγράμματα Ροής (flowchart) για την απεικόνιση της επιχειρηματικής λογικής καθώς και τα διαγράμματα του Σχεσιακού Μοντέλου (Entity-Relation Diagrams) για την αναπαράσταση της Βάσης Δεδομένων, διασφαλίζεται η δημιουργία μια λειτουργικής και αποδοτικής πλατφόρμας. Επιπλέον με την εφαρμογή προηγμένων διεπαφών χρήστη UI (User Interface) και μηχανών αναζήτησης εξασφαλίζεται η βελτιωμένη εμπειρία για τους θεατές κατά τη διάρκεια της κράτησης εισιτηρίων και αποτελεσματικής διαχείρισης των εκδηλώσεων για τους διοργανωτές.

1.4 Οργάνωση Κεφαλαίων

Τα υπόλοιπα κεφάλαια της παρούσας εργασίας, οργανώνονται όπως ακολούθως:

Στο Κεφάλαιο 2 αναλύονται οι βασικές έννοιες, οι αρχές που σχετίζονται με το πρόβλημα της διαδικτυακής πλατφόρμας διαχείρισης κρατήσεων σε πολιτιστικές εκδηλώσεις.

Στο Κεφάλαιο 3 παρουσιάζονται και αναλύονται συναφείς με την παρούσα εργασία έρευνες, σε αντιδιαστολή με την παρούσα εργασία, και τα κενά που αυτή συμπληρώνει.

Το Κεφάλαιο 4 περιλαμβάνει την παρουσίαση της μεθοδολογίας που ακολουθήθηκε για την ανάπτυξη της εφαρμογής, και τα δεδομένα που χρησιμοποιήθηκαν για την επίδειξη (demo).

Στο Κεφάλαιο 5 παρουσιάζεται η υλοποίηση της εφαρμογής και η αξιολόγηση της απόδοσης μέσω δοκιμών και αξιολογήσεων.

Στο Κεφάλαιο 6 περιλαμβάνεται η παρουσίαση της εφαρμογής μέσα από στιγμιότυπα κατά την λειτουργία της demo έκδοσης της εφαρμογής.

Το Κεφάλαιο 7 περιλαμβάνει τα συμπεράσματα για τα αποτελέσματα της εργασίας και προτείνονται μελλοντικές επεκτάσεις και βελτιώσεις.

Κεφάλαιο 2ο: Θεωρητικό υπόβαθρο

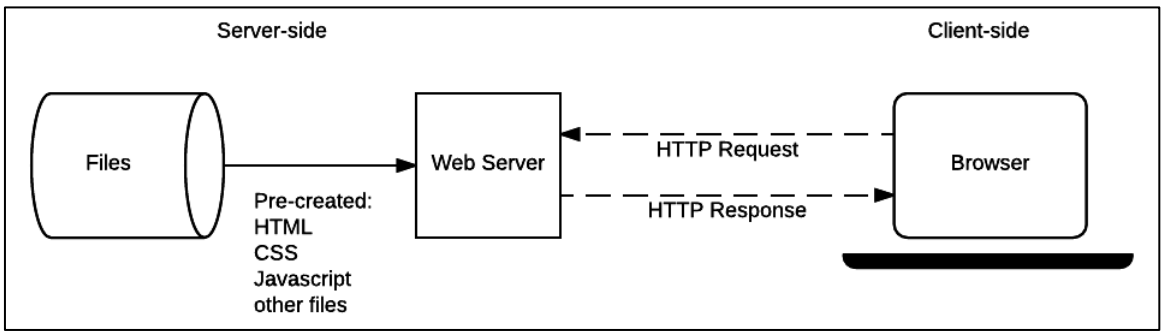
Στο κεφάλαιο αυτό, αναλύεται το θεωρητικό υπόβαθρο που στηρίζει την παρούσα εργασία και παρουσιάζονται οι βασικές έννοιες και αρχές που σχετίζονται με το πρόβλημα της εφαρμογής διαχείρισης κρατήσεων για πολιτιστικές εκδηλώσεις.

Συγκεκριμένα, εξετάζονται τα διάφορα τεχνολογικά εργαλεία και πακέτα ανάπτυξης (frameworks) που χρησιμοποιούνται στην ανάπτυξη της εφαρμογής, συμπεριλαμβανομένων των Angular, Nest.JS και MongoDB. Εξετάζονται οι τρόποι με τους οποίους μπορούν να εφαρμοστούν συστήματα διαχείρισης χρηστών για την εξυπηρέτηση διαφορετικών ρόλων όπως οι διοργανωτές και οι θεατές. Τέλος, εξετάζεται η σημασία του σχεδιασμού και της υλοποίησης διεπαφών χρήστη (UI) που προσφέρουν εύκολη πλοήγηση και βελτιωμένη εμπειρία χρήστη.

Τα παραπάνω frameworks, οικειοποιούνται μια καινούρια γλώσσα-υπερσύνολο της γλώσσας JavaScript, η οποία ονομάζεται TypeScipt[1], και δημιουργήθηκε για να συμπληρώσει την ανάπτυξη script κώδικα συμπεριλαμβάνοντας επιπλέον δυνατότητες, δίνοντας έμφαση στον ορισμό των κλάσεων και αντικειμένων που χρησιμοποιεί με τη χρήση τύπων (types).

2.1 Στατικές και Δυναμικές σελίδες

Οι στατικές σελίδες περιλαμβάνουν στοιχειώδεις HTML (HyperText Markup Language) σελίδες, των οποίων το περιεχόμενο έχει καθοριστεί κατά την ανάπτυξη τους και παραμένει αμετάβλητο για οποιονδήποτε χρήστη. Όπως απεικονίζεται και στο σχεδιάγραμμα της Εικόνας 2.1, οποτεδήποτε ο χρήστης πλοηγηθεί στη σελίδα (πληκτρολογώντας το URL στον φυλλομετρητή) ο φυλλομετρητής στέλνει ένα HTTP (HyperText Transfer Protocol) “GET“ αίτημα για να λάβει μια νέα σελίδα από το διακομιστή [1]. Ο διακομιστής θα απαντήσει είτε με HTTP κωδικό κατάστασης (HTTP status code) διακόσια (200) σε περίπτωση επιτυχούς αποστολής της σελίδας, είτε με κάποιον διαφορετικό κωδικό που υποδηλώνει σφάλμα (πχ. 500 για την περίπτωση σφάλματος διακομιστή).

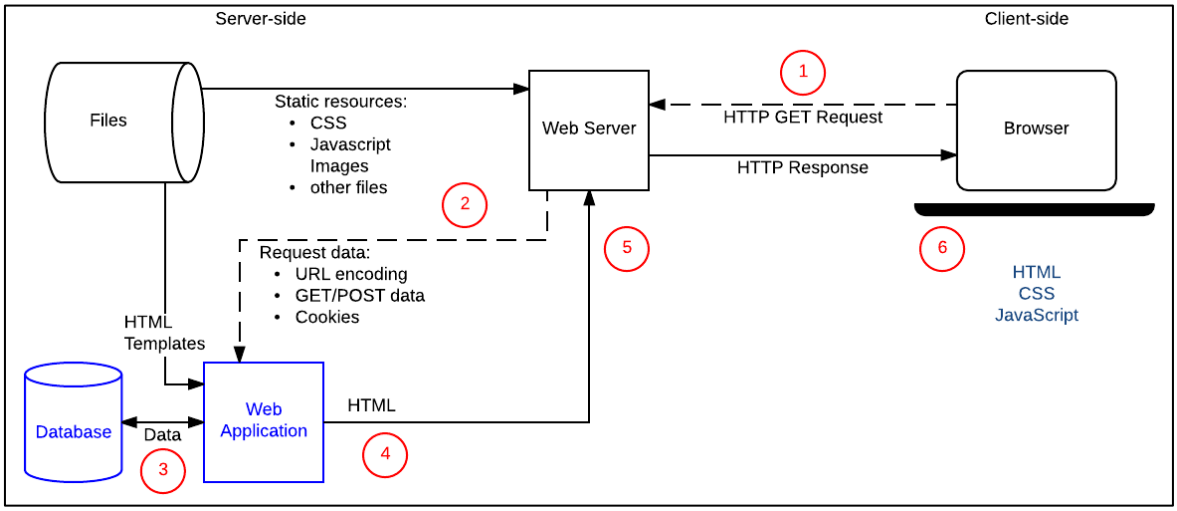


Εικόνα 2. 1: Αρχιτεκτονική Στατικής Σελίδας [1].

Αντίθετα στις δυναμικές σελίδες, ο διακομιστής είναι εκείνος που δημιουργεί το περιεχόμενο της σελίδας και είτε μπορεί να τροποποιηθεί από τους χρήστες του φυλλομετρητή είτε να δημιουργηθεί σύμφωνα με τις οδηγίες (με τη μορφή κώδικα) που του έχουν παρασχεθεί. Συνήθως, το περιεχόμενο που παράγεται από το επίπεδο του διακομιστή «εκχύεται» (injected) σε HTML πρότυπα (templates) που έχουν προετοιμαστεί προηγουμένως. Όπως παρουσιάζεται στο διάγραμμα της Εικόνας 2.2, αφού ο φυλλομετρητής αποστείλει ένα HTTP ‘GET’ αίτημα στο διακομιστή, εκείνος θα απαντήσει

[1] https://www.typescriptlang.org/

ως στατική σελίδα εάν η απάντηση αφορά στατικό περιεχόμενο (πχ. PDF αρχείο), ενώ σε διαφορετική περίπτωση θα προηγηθεί επικοινωνία με τη Βάση Δεδομένων για την ανάκτηση περιεχομένου και θα επιστρέψει απάντηση με το περιεχόμενο αυτό.

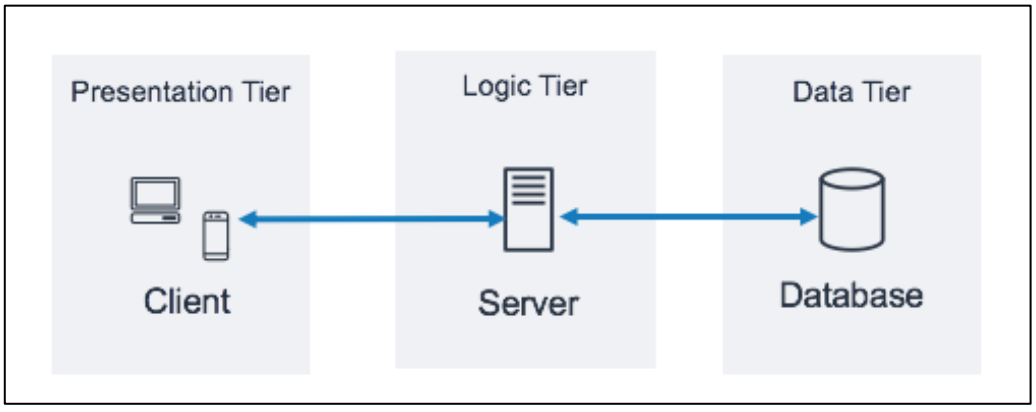


Εικόνα 2. 2: Αρχιτεκτονική Δυναμικής Σελίδας [1].

Η χρήση της μίας ή της άλλης περίπτωσης, υπό κανονικές συνθήκες είναι απόφαση που έγκειται στις απαιτήσεις του εκάστοτε οργανισμού, παρά το γεγονός πώς η χρήση των στατικών ιστοσελίδων έχει φθίνει στις μέρες μας.

2.2 Three-tier Αρχιτεκτονική

Η αρχιτεκτονική πολλαπλών επιπέδων (multi-tier architecture) αποτελεί ένα διαδεδομένο τρόπο οργάνωσης μιας εφαρμογής, ώστε να διαχωρίζονται σε επιμέρους επίπεδα τα λειτουργικά και τα φυσικά της τμήματα. Στο πλαίσιο της παρούσας εργασίας, όπως φαίνεται και στην Εικόνα 2.3 υιοθετείται μια πολλαπλών επιπέδων αρχιτεκτονική με τρία κύρια επίπεδα (three-tier architecture): το επίπεδο πελάτη (frontend), το επίπεδο διακομιστή (backend) και το επίπεδο βάσης δεδομένων (database).



Εικόνα 2. 3: Αρχιτεκτονική 3 επιπέδων [2]

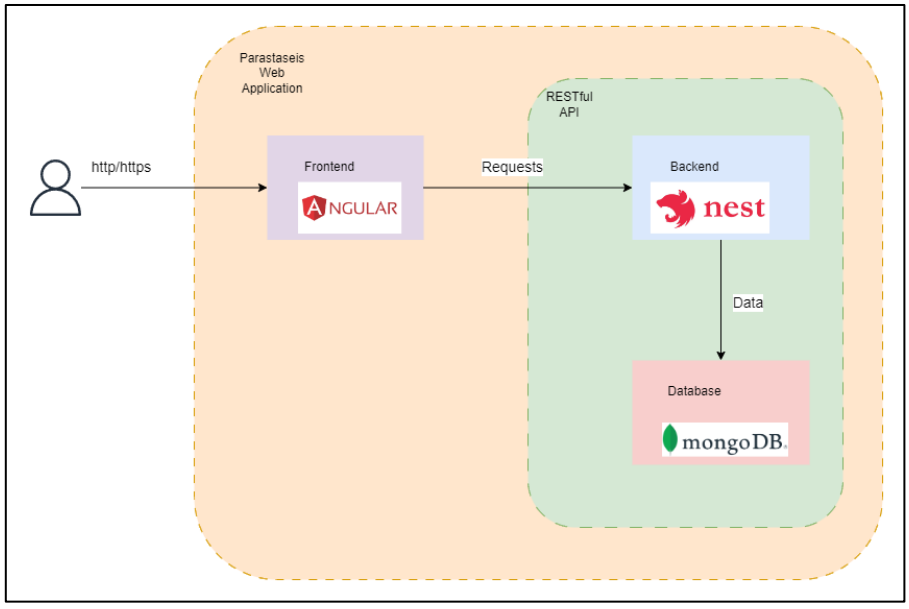
Το επίπεδο Πελάτη (frontend), αποτελεί το τμήμα εκείνο της εφαρμογής με το οποίο αλληλοεπιδρούν οι χρήστες. Συνήθως πρόκειται για την ιστοσελίδα ή την εφαρμογή που τρέχει στο φυλλομετρητή (browser) του χρήστη. Σε αυτό το επίπεδο υλοποιούνται οι διεπαφές χρήστη (UI), οι οποίες επιτρέπουν στους χρήστες να αλληλοεπιδράσουν με την εφαρμογή. Σε κάθε ρόλο χρήστη

υπάρχει δυνατότητα να αντιστοιχεί μια διεπαφή, η οποία παρέχει και διαφορετικό σύνολο δυνατοτήτων αλληλεπίδρασης με την εφαρμογή.

Το επίπεδο Διακομιστή (backend), αναλαμβάνει την επεξεργασία των αιτήσεων που αποστέλλονται από το επίπεδο πελάτη και συνεπώς από το χρήστη. Εδώ βρίσκονται οι κανόνες επιχειρηματικής λογικής (business logic) της εφαρμογής και οι διάφορες λειτουργίες. Το επίπεδο αυτό αναλαμβάνει να διαχειρίζεται τα δεδομένα και να αλληλοεπιδρά με το επίπεδο βάσης δεδομένων.

Τέλος, το επίπεδο Βάσης Δεδομένων, αποθηκεύει και διαχειρίζεται τα δεδομένα που χρησιμοποιεί η εφαρμογή. Εδώ αποθηκεύονται πληροφορίες κοινωνικών εκδηλώσεων (όπως θεατρικές παραστάσεις), εισιτηρίων και κρατήσεων. Το επίπεδο αυτό αναλαμβάνει την αποτελεσματική αποθήκευση, ανάκτηση και ενημέρωση των δεδομένων. Τα προβλήματα με τα οποία ασχολείται αυτό το επίπεδο αφορούν την αποτελεσματική αποθήκευση, την ταχύτητα προσπέλασης και ανάκτησης, και την εξοικονόμηση αποθηκευτικού χώρου.

Η παραπάνω αρχιτεκτονική τριών επιπέδων, με τη χρήση της στοίβας MEAN, εφαρμόζεται όπως παρουσιάζεται στην Εικόνα 2.4. Πιο συγκεκριμένα, στο επίπεδο πελάτη συναντάμε την «Angular[2]», με την οποία αλληλοεπιδρά ο χρήστης αξιοποιώντας έναν φυλλομετρητή (browser). Παρότι η Angular ανήκει στα πακέτα ανάπτυξης (frameworks) υποστήριξης Εφαρμογών Μονής Σελίδας [3], είναι εξαιρετικά αποδοτική και στο ρόλο της στην υπόψη αρχιτεκτονική. Στο επίπεδο Διακομιστή το ρόλο του «διαμεσολαβητή» μεταξύ των υπόλοιπων δύο επιπέδων αναλαμβάνει η «Nest.JS[3]», η οποία έχει προεπιλεγμένες μεθόδους διαχείρισης HTTP αιτημάτων, και μια σειρά από οδηγούς (drivers) για την ευκολότερη επικοινωνία με το τρίτο επίπεδο της Βάσης Δεδομένων.



Εικόνα 2. 4: Υλοποίηση 3-tier αρχιτεκτονικής με τη χρήση της στοίβας MEAN.

Στο επίπεδο της Βάσης Δεδομένων, η «MongoDB[4]» αναλαμβάνει την αποτελεσματική αποθήκευση και προσπέλαση των δεδομένων, έχοντας προβάδισμα έναντι των άλλων τύπων Βάσεων Δεδομένων στη διαχείριση Μεγάλων Δεδομένων [4] και τα πεδία της Στατιστικής και Ανάλυσης Δεδομένων

[2] https://angular.io/
[3] https://nestjs.com/
[4] https://www.mongodb.com/

[5]. Για την επικοινωνία με τη MongoDB, η Nest.JS αξιοποιεί τη ODM (Object Data Modeling) βιβλιοθήκη «Mongoose[5]», η οποία είναι ευρέως διαδεδομένη ανάμεσα σε όλες τις γλώσσες προγραμματισμού για το επίπεδο διακομιστή (backend).

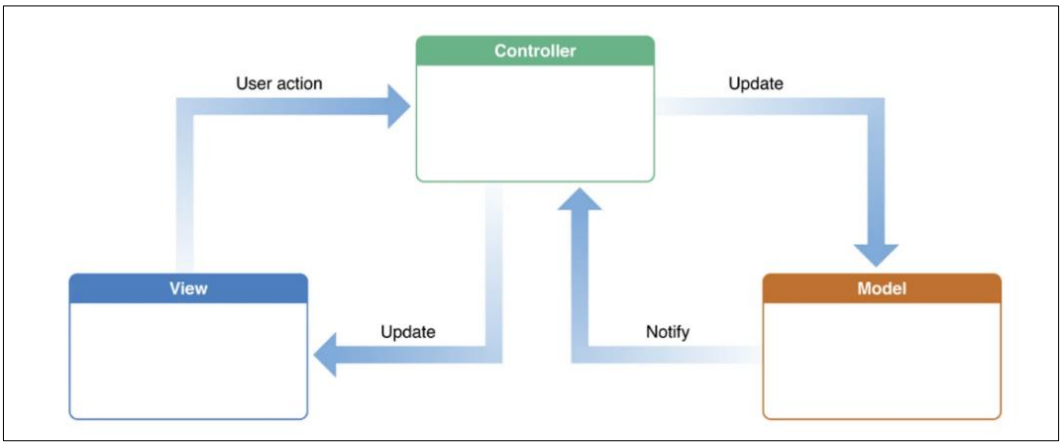
Με την αρχιτεκτονική αυτή, επιτρέπεται ο διαχωρισμός των αρμοδιοτήτων και των λειτουργιών της εφαρμογής, προσφέροντας καλύτερη οργάνωση και συντήρηση. Κάθε επίπεδο είναι υπεύθυνο για συγκεκριμένες αρμοδιότητες και αλληλοεπιδρά με τα άλλα επίπεδα για να αποτελέσει μια ενιαία λειτουργική πλατφόρμα διαχείρισης κρατήσεων.

2.3 Model-View-Controller (MVC) Αρχιτεκτονική

Η Model-View-Controller (MVC) αρχιτεκτονική αποτελεί ένα πρότυπο οργάνωσης του κώδικα μιας εφαρμογής που στοχεύει στο διαχωρισμό της επιχειρηματικής λογικής (business logic) από τη διεπαφή (UI) του χρήστη. Αυτό ενισχύει την απομόνωση των διαφορετικών τμημάτων της εφαρμογής για λόγους συντήρησης, επεκτασιμότητας και εκσυγχρονισμού.

Η αρχιτεκτονική αυτή αποτελείται από τρία βασικά τμήματα όπως φαίνεται στην Εικόνα 2.5, τα οποία της έχουν δώσει και το όνομα της:

1. Το Μοντέλο (Model), το οποίο αντιπροσωπεύει τα δεδομένα της εφαρμογής και την λειτουργικότητα των διάφορων μεθόδων της επιχειρηματικής λογικής της εφαρμογής. Αυτό το τμήμα είναι υπεύθυνο για την ανάκτηση, αποθήκευση και επεξεργασία των δεδομένων, καθώς και για την υλοποίηση των μεθόδων που συνιστούν την επιχειρηματική λογική της εφαρμογής.
2. Η προβολή (View), αναλαμβάνει την απεικόνιση των δεδομένων που παρέχονται από το τμήμα του Μοντέλου στους χρήστες. Πρόκειται ουσιαστικά, για το τμήμα που απεικονίζει τις πληροφορίες, που παρέχονται από το Μοντέλο σύμφωνα με τις οδηγίες που δίνονται από το επόμενο τμήμα που ονομάζεται Ελεγκτής (Controller).
3. Ο Ελεγκτής (Controller), διαχειρίζεται τις ενέργειες του χρήστη και επικοινωνεί με το Μοντέλο για την αποστολή και λήψη δεδομένων. Ανταποκρίνεται στα αιτήματα του χρήστη, που παρέχονται μέσω της Προβολής, και ανακατευθύνει τις κατάλληλες ενέργειες, επιτρέποντας την ενημέρωση του Μοντέλου και την εμφάνιση των αποτελεσμάτων στην Προβολή.



Εικόνα 2. 5: MVC αρχιτεκτονική [6]

[5] https://mongoosejs.com/

Η αρχιτεκτονική MVC χρησιμοποιείται για την απομόνωση των ανεξάρτητων τμημάτων της εφαρμογής, συμβάλλοντας στην ευκολότερη διαχείριση, συντήρηση και επέκταση της. Το σημαντικότερο όμως πλεονέκτημα είναι πως είναι συμβατή με την αρχιτεκτονική πολλαπλών επιπέδων (Multi-tier architecture) όπως η ανάπτυξη μιας εφαρμογής διαχείρισης κρατήσεων πολιτιστικών εκδηλώσεων.

2.4 MVC Αρχιτεκτονική και στοίβα MEAN

Η αρχιτεκτονική MVC αποτελεί τη βασική αιτία πίσω από την ανάγκη δημιουργίας την Angular ως πακέτο (framework) ανάπτυξης κώδικα και κατευθυντήρια γραμμή στους προγραμματιστές που τη χρησιμοποιούν.

Πιο συγκεκριμένα, για την Angular:

1. Το Μοντέλο (Model) αφορά τα δεδομένα της εφαρμογής και σε μεγάλο βαθμό λαμβάνονται από το επίπεδο Διακομιστή (backend). Υπάρχει δυνατότητα μεταποίησης των δεδομένων από τη στιγμή που θα ληφθούν από την Angular, όπως και πριν αποσταλούν στο Διακομιστή γεγονός που της αποδίδει και το ρόλο του Μοντέλου. Συνήθως οι μέθοδοι του εντοπίζονται στα αρχεία επέκτασης «.component.ts» που συνοδεύουν κάθε αρχείο του επιπέδου Προβολής (View).
2. Η Προβολή (View) αναπαριστά τις διεπαφές (UI) των χρηστών. Ο κώδικας συνήθως αφορά το πρότυπο (template) και συναντάται στα αρχεία επέκτασης «.component.html», και παρέχει την όψη των δεδομένων και των στοιχείων του UI.
3. Ο Ελεγκτής (Controller) είναι υπεύθυνος για την υλοποίηση της λογικής επεξεργασίας των δεδομένων και των ενεργειών που λαμβάνονται από τους χρήστες και συνήθως συναντάται σε αρχεία επέκτασης «.service.ts».

Ομοίως για τη Nest.JS, η οποία αποτελεί την συμμόρφωση της Express.js στην αρχιτεκτονική MVC, διακρίνουμε τα παρακάτω μέρη:

1. Το Μοντέλο (Model) αντιπροσωπεύει τις Δομές Δεδομένων ή Σχήματα της Βάσης Δεδομένων και την επικύρωση τους πριν την αποθήκευση τους στη Βάση Δεδομένων και συναντάται συνήθως σε αρχεία επέκτασης «.model.ts».
2. Ο Ελεγκτής (Controller) γεφυρώνει τα μοντέλα και τις υπηρεσίες, διαχειριζόμενος τα HTTP (Hypertext Transfer Protocol) αιτήματα που καταφθάνουν από το επίπεδο Πελάτη (frontend) και συγκεκριμένα την Angular. Εδώ επεξεργάζονται τα αιτήματα, γίνεται επικοινωνία με το Μοντέλο και επιστρέφονται τα αποτελέσματα στο frontend. Συναντάται συνήθως σε αρχεία επέκτασης «.controller.ts»
3. Η Υπηρεσία (Service), έχει την ευθύνη υλοποίησης της επιχειρηματικής λογικής της εφαρμογής, περιλαμβάνοντας τις κλήσεις προς τη Βάση Δεδομένων, τους κανόνες λογικής και άλλες μεθόδους επεξεργασίας δεδομένων. Συναντάται συνήθως σε αρχεία επέκτασης «.service.ts»

Ως προς τη χρήση της MongoDB, αυτή συμπληρώνει το κομμάτι του Μοντέλου (Model) αποθηκεύοντας τα δεδομένα σε συλλογές (collections) που αποθηκεύονται ως αρχεία BSON (Binary JSON) στο δίσκο και μπορούν να περιγράφουν με αντικείμενα (objects) JavaScript. Η χρήση της αποδεικνύεται αποδοτικότερη όταν απαιτείται ταχύτερη προσπέλαση των δεδομένων καθώς και όταν το μέγεθος των δεδομένων προσεγγίζει αυτό των Μεγάλων Δεδομένων (Big Data). Το τελευταίο την καταστά ιδανική επιλογή για παραγωγή στατιστικών και ανάλυσης όπου απαιτείται.

2.5 Συστήματα Διαχείρισης Χρηστών

Ένα από τα πιο θεμελιώδη ζητήματα, για την ασφαλή λειτουργία μιας εφαρμογής είναι η πιστοποίηση και η εξουσιοδότηση των χρηστών. Εάν πρόκειται περί μιας εφαρμογής η οποία παρέχει διαφορετικές δυνατότητες ανά ρόλο χρήστη, όπως μια εφαρμογή διαχείρισης κρατήσεων εισιτηρίων κοινωνικών εκδηλώσεων, στην οποία δε θα μπορούν οι θεατές να επεξεργαστούν τις πληροφορίες μιας εκδήλωσης, τότε καθίσταται επιτακτική η επιβολή περιορισμών με βάση τη πιστοποίηση του χρήστη.

Για συστήματα που εξυπηρετούν μεγάλους οργανισμούς εκατοντάδων χρηστών, έχουν αναπτυχθεί εξειδικευμένα συστήματα/εφαρμογές για τη διαχείριση των χρηστών τους. Μερικά παραδείγματα τέτοιων συστημάτων ανοιχτού κώδικα είναι τα OpenLDAP[6], Apache Directory Server[7], και FreeIPA[8], τα οποία αποτελούν ολοκληρωμένες λύσεις διαχείρισης χρηστών με βάση ορισμένους ρόλους όπως διαχειριστής (administrator), διαχειριστής τομέα (domain admin) και χρήστης (user).

Σε εφαρμογές μικρότερης κλίμακας, όπου δεν υπάρχουν πολλοί ρόλοι, η διαχείριση των χρηστών είναι πιο απλή και μπορεί να εφαρμοστεί με απλούστερες λύσεις στον ίδιο τον κώδικα του διακομιστή (backend). Μια από τις πλέον διαδεδομένες τεχνολογίες διαχείρισης σύγχρονης πιστοποίησης είναι η JWT (Json Web Token), η οποία παρέχεται από τις προεπιλεγμένες βιβλιοθήκες της Nest.JS. Σύμφωνα με το documentation του JWT [6] «*Το JWT είναι ένα πρότυπο ανοιχτού κώδικα (RFC 7519) το οποίο ορίζει ένα συμπαγές και αυτόνομο τρόπο για τη μετάδοση πληροφορίας μεταξύ συμβαλλόμενων μελών με τη μορφή JSON αντικειμένου. Αυτή η πληροφορία μπορεί να πιστοποιηθεί καθώς είναι ψηφιακά υπογεγραμμένη*». Η υπογραφή αυτή μπορεί να υλοποιηθεί είτε με τη χρήση ενός μόνο κλειδιού (συμμετρική κρυπτογραφία) είτε με τη χρήση ιδιωτικού/δημοσίου κλειδιού (ασύμμετρη κρυπτογραφία RSA ή ECDSA αλγορίθμου).

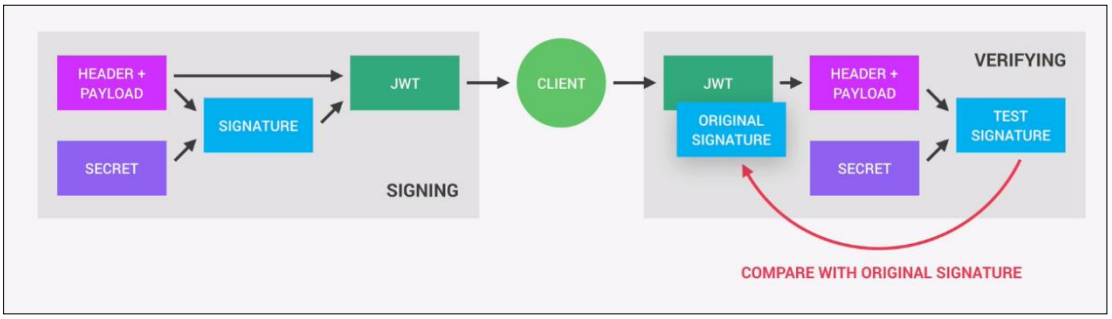
Το JWT λειτουργεί αξιοποιώντας ένα κουπόνι (token), το οποίο αποτελείται από τρία συστατικά μέρη: την κεφαλίδα (header), το ωφέλιμο φορτίο (payload), και την υπογραφή (signature). Η κεφαλίδα περιλαμβάνει τις πληροφορίες για το κουπόνι, όπως τον τύπο του κουπονιού, η οποία είναι από προεπιλογή 'JWT' και τον αλγόριθμο κρυπτογράφησης, ο οποίος είναι 'sha256' από προεπιλογή. Το ωφέλιμο φορτίο περιλαμβάνει τα δεδομένα που αποστέλλονται στον διακομιστή μέσω του JWT, και τέλος η υπογραφή προκύπτει από τη σύνθεση της κεφαλίδας με το ωφέλιμο φορτίο. Με τον τρόπο αυτό μπορεί να εξακριβωθεί η ταυτότητα του αποστολέα και να εξασφαλισθεί ο εντοπισμός αλλοίωσης του κουπονιού [6].

Πιο συγκεκριμένα , όπως φαίνεται στην Εικόνα 2.6, όπου διακρίνονται οι ενέργειες τόσο από την πλευρά του χρήστη όσο και από την πλευρά του διακομιστή, η πιστοποίηση στηρίζεται στη σύγκριση της υπογραφής που αποστέλλεται από το χρήστη, με την υπογραφή που παράγει ο ίδιος ο διακομιστής, και σε περίπτωση που αυτές δεν ταιριάζουν απορρίπτει το κουπόνι καθώς θεωρείται ότι έχει παραβιαστεί [7].

Με τη χρήση του JWT, είναι πρακτικά αδύνατη η αλλαγή του κουπονιού χωρίς να επηρεαστεί η υπογραφή του, και επομένως η μη εξουσιοδοτημένη πρόσβαση στην εφαρμογή μας. Αξιοποιώντας αυτή τη συμπεριφορά, προκειμένου να εξασφαλιστεί η εξουσιοδότηση του χρήστη αποστέλλεται σε αυτόν ένα JWT κουπόνι με την είσοδο του (login) στην εφαρμογή, αρχικοποιώντας μια συνεδρία (session) στον διακομιστή. Σε κάθε επόμενη σελίδα που επισκέπτεται ο χρήστης αποστέλλει το

[6] https://www.openldap.org/
[7] https://directory.apache.org/
[8] https://www.freeipa.org/

κουπόνι αυτό για να πιστοποιηθεί η εξουσιοδότηση του στη συγκεκριμένη σελίδα.



Εικόνα 2. 6: Υπογραφή και Εξακρίβωση JWT [8].

Με τον τρόπο αυτό, και συγκεκριμένα με την επίδειξη του token του, εξακριβώνεται η εξουσιοδότηση του χρήστη σε κάθε σελίδα της εφαρμογής.

2.6 RESTful Εφαρμογή

Μια τάση που συναντάται σε πληθώρα εφαρμογών στο Διαδίκτυο είναι οι εφαρμογές REST. Μια RESTful εφαρμογή ακολουθεί τις αρχές του REST (Representation State Transfer) για το σχεδιασμό και την ανάπτυξη του δικτυακού περιβάλλοντος της. Πρόκειται περί αρχιτεκτονικής που περιγράφει τον τρόπο αλληλεπίδρασης μεταξύ συσκευών και εφαρμογών στο διαδίκτυο.

Οι βασικές αρχές του RESTπεριλαμβάνουν τα κάτωθι [8]:

- Οι διάφορες οντότητες που είναι προσβάσιμες μέσω της εφαρμογής (π.χ. χρήστες, προϊόντα, κρατήσεις) αντιμετωπίζονται ως **πόροι**. Κάθε πόρος έχει μια μοναδική URI που τον διακρίνει μοναδικά.
- **Οι πόροι αναπαρίστανται με δεδομένα**, συνήθως σε μορφή JSON ή XML. Σε περιβάλλον πελάτη-εξυπηρετητή, οι πελάτες ζητάνε τα δεδομένα και τα «περιμένουν» (δλδ. η εξέλιξη του κώδικα βασίζεται) στις παραπάνω μορφές.
- Οι αιτήσεις προς τον εξυπηρετητή δε βασίζονται σε καμία προηγούμενη κατάσταση ή συνεδρία (session) και είναι **αυτοτελείς/ανεξάρτητες (stateless)**.
- Η **διεπαφή (Uniform Interface)** μεταξύ του πελάτη και του εξυπηρετητή θα πρέπει να είναι καθορισμένη, ομοιόμορφη και κατανοητή. Οι αλληλεπιδράσεις γίνονται με τη χρήση HTTP αιτημάτων (requests) και κυρίως των GET, POST, PUT, DELETE.
- Οι πόροι πρέπει να διατηρούν **συνοχή και συνεκτικότητα (Consistency).** Οι αλλαγές σε έναν πόρο πρέπει να είναι ορατές από όλους τους χρήστες που αναφέρονται σε αυτόν.

Ο σχεδιασμός μιας εφαρμογής ως RESTful παρέχει αρκετά πλεονεκτήματα, μερικά από τα οποία είναι η ευελιξία στην ανάπτυξη, απλότητα στην κατανόηση των αλληλεπιδράσεων μεταξύ των πόρων, και απομόνωση των μεθόδων που συνεπάγεται στην ευκολία ανακατασκευής του δικτύου χωρίς να επηρεάζονται άλλες υπάρχουσες λειτουργίες.

2.7 MongoDB

Η MongoDB είναι μια δημοφιλής Βάση Δεδομένων ανοιχτού κώδικα, η οποία σχεδιάστηκε ώστε να διαχειρίζεται αδόμητα ή ημι-δομημένα δεδομένα μεγάλου όγκου. Η κύρια διαφορά της από τις παραδοσιακές Βάσεις Δεδομένων σχετίζεται με την παροχή ενός πιο ευέλικτου και δυναμικού τρόπου αποθήκευσης και ανάκτησης των δεδομένων.

Τα κύρια χαρακτηριστικά της σύμφωνα με τη βιβλιογραφία της [9], είναι τα παρακάτω:

- Το βασικό της δομικό στοιχείο είναι τα **Documents**, στα οποία αποθηκεύονται τα δεδομένα σε μορφή που έχει ομοιότητες με τα αρχεία JSON, γνωστά ώς BSON (Binary JSON). Κάθε document μπορεί να έχει τη δικιά του δομή, επιτρέποντας έτσι διαφορετικούς τύπους δεδομένων μέσα στη Βάση Δεδομένων, η οποία μπορεί να αλλάξει οποιαδήποτε στιγμή αφού δεν περιορίζεται από κάποιο σχήμα.
- Τα documents οργανώνονται σε **collections**, τα οποία αντιστοιχούν στους πίνακες της Σχεσιακής Βάσης Δεδομένων. Τα collections δεν είναι απαραίτητο να περιλαμβάνουν documents ιδίας δομής, επιτρέποντας με αυτό τον τρόπο την αποθήκευση ενός document εντός άλλου document, το οποίο αναφέρεται στο πρώτο.
- Παρέχεται η δυνατότητα **τροποποίησης του σχήματος** ενός document, ακόμα κι αν χρησιμοποιείται μετά τη δημιουργία του.
- Διαθέτει δική του **γλώσσα επερωτήσεων**, η οποία περιλαμβάνει δυνατότητες όπως φιλτράρισμα (filtering), ταξινόμησης (sorting), συσσώρευση (aggregation), ακόμα και γεωχωρικά επερωτήματα (geospatial queries).
- Περιλαμβάνει ένα framework συσσώρευσης (**aggregation framework**), το οποίο προσφέρει επιπλέον δυνατότητες τροποποίησης των δεδομένων (data transformation), προσφέροντας ανάλυση με επερωτήματα ομαδοποίησης (grouping), φιλτραρίσματος (filtering), και υπολογίζοντας συγκεντρωτικές πράξεις (aggregate values).
- Διαθέτει τη δυνατότητα υψηλής διαθεσιμότητας **(High Availability**), προσφέροντας συνεργασία με αντίγραφα (replicas) του εαυτού του, για διατήρηση των δεδομένων σε πολλά σημεία.
- Διαθέτει δυνατότητα τεμαχισμού μεγάλων δομών της (**sharding**), διαμοιράζοντας τα δεδομένα μεταξύ πολλαπλών διακομιστών με βάση το κλειδί (sharding key) για το χειρισμό μεγάλων δεδομένων.
- Υπάρχει δυνατότητα δοσοληψιών (**transactions**) στα πρότυπα των σχεσιακών Βάσεων Δεδομένων.
- Δίνεται η δυνατότητα εκτέλεσης **επερωτήσεων σε μορφή JSON**, για μεγαλύτερη ακρίβεια στις απαντήσεις (query responses).
- Διατίθεται δυνατότητα εκτέλεσης γεωχωρικών επερωτήσεων (**geospatial queries**).
- Και δύναται η διατήρηση ιστορικότητας-εκδόσεων (**document versioning**) των documents, για την παρακολούθηση των αλλαγών ενός document.

Καταλήγοντας, η MongoDB διαθέτει μια σειρά από πλεονεκτήματα τα οποία αξιοποιούνται συνήθως από εφαρμογές των οποίων τα δεδομένα αναμένεται να αλλάζουν τακτικά, όπως οι εφαμογές CMS (Content Management Systems) [10], εφαρμογές διαδικτυακών πωλήσεων (e-commerce), παροχή στατιστικών αναλύσεων και εφαρμογών κινητών συσκευών.

2.8 Mongoose

Όπως αναφέραμε στο προηγούμενο Κεφάλαιο η MongoDB είναι μια NoSQL Βάση Δεδομένων, η οποία αποθηκεύει τα δεδομένα σε αρχεία επέκτασης BSON (Binary JSON), τα οποία δε διαθέτουν σχήμα για τη Βάση Δεδομένων. Για να εισάγουμε σχήμα και δομή σε μια κατά τα άλλα μη-δομημένη και χωρίς σχήμα Βάση Δεδομένων χρησιμοποιούμε μία Object Data Modeling (ODM) βιβλιοθήκη σε ένα επίπεδο πάνω από τη MongoDB, με την ονομασία Mongoose.

Τα κύρια χαρακτηριστικά της Mongoose σύμφωνα με τη βιβλιογραφία της [11], είναι:

- το **Σχήμα (Schema)**, το οποίο ορίζει τη δομή των documents ενός collection. Ορίζει συγκεκριμένα τα πεδία, τους τύπους δεδομένων τους, τις προεπιλεγμένες τιμές τους, καθώς και την εγκυρότητα τους (εάν δλδ. περνάνε επιτυχώς τους ελέγχους που ορίζουμε εμείς),
- το **Μοντέλο (Model)**, το οποίο αποτελεί ένα αντικείμενο, το οποίο αντιπροσωπεύει ένα Collection της Βάσης Δεδομένων, και με το οποίο μπορεί να αλληλοεπιδράσει η Nest.JS. Με άλλα λόγια αποτελεί μια διεπαφή ενός collection της Βάσης Δεδομένων, το οποίο προσφέρει CRUD (Create, Read, Update, Delete) δυνατότητες και μας επιτρέπει να ορίσουμε μεθόδους.
- Η **Εγκυρότητα (Validation)**, ως μέσο εξακρίβωσης εγκυρότητας των δεδομένων που προορίζονται να αποθηκευτούν στη Βάση Δεδομένων, αποτρέποντας έτσι την αποθήκευση των δεδομένων εκείνων που δεν ανταποκρίνονται στο ορισμένο σχήμα του Collection στο οποίο αντιστοιχούν.
- Οι **Ενδιάμεσες Μέθοδοι (Middlewares)**, είναι συναρτήσεις οι οποίες επιτρέπουν την εκτέλεση κώδικα πριν ή μετά την εκτέλεση συγκεκριμένων ενεργειών, όπως για παράδειγμα την αποθήκευση ενός document στη Βάση Δεδομένων.
- Η **Συμπλήρωση (Population)**, αφορά τη συμπλήρωση πεδίων ενός document τα οποία αναφέρονται σε άλλα documents πριν την αποστολή του στον χρήστη.

Μεταξύ πολλών γλωσσών προγραμματισμού, από τις οποίες υποστηρίζεται και με τις οποίες συνεργάζεται αρμονικά η Mongoose είναι και το framework της Nest.JS, το οποίο είναι ιδανικό για την ανάπτυξη επεκτάσιμων και συντηρήσιμων Node.JS εφαρμογών.

2.9 Node.JS

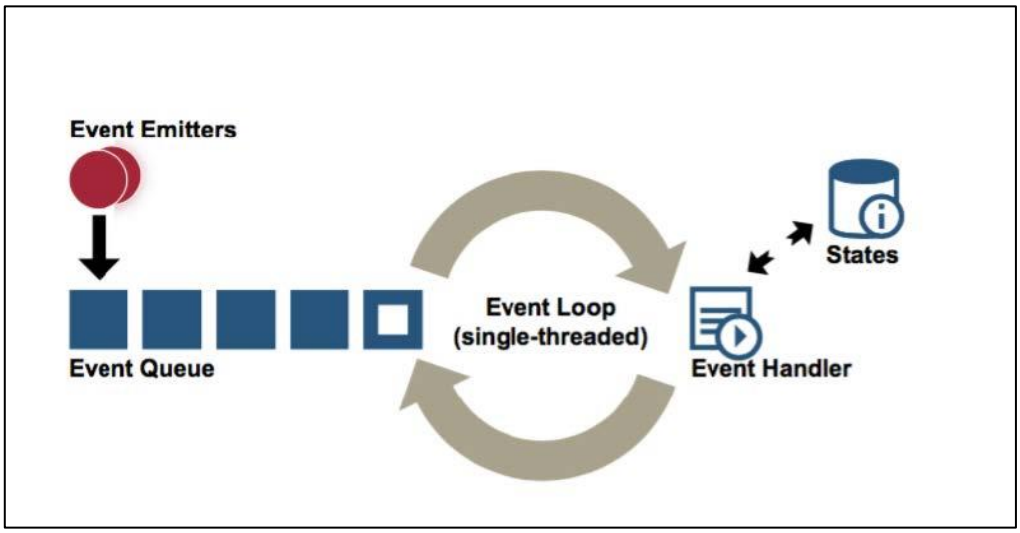
Το Node.JS είναι ένα περιβάλλον ανοιχτού-κώδικα, για την εκτέλεση JavaScript κώδικα στο επίπεδο του Διακομιστή, το οποίο επιτρέπει στους διαχειριστές την ανάπτυξη επεκτάσιμων και υψηλών επιδόσεων διαδικτυακών εφαρμογών. Έχει χτιστεί επάνω στο V8 JavaScript engine, στο οποίο βασίζει και την λειτουργία του ο φυλλομετρητής Google Chrome. Έχει αναπτυχθεί με γνώμονα την ανάπτυξη ασύγχρονων, και ακολουθούμενων από γεγονότα (event-driven) εφαρμογών, καθιστώντας το, ιδανικό για την ανάπτυξη εφαρμογών που απαιτούν άμεση αλληλεπίδραση, ταυτόχρονη εκτέλεση και αποτελεσματική διαχείριση λειτουργιών Εισόδου/Εξόδου (I/O, Input/Output) [12].

Τα κύρια χαρακτηριστικά του, σύμφωνα με τη βιβλιογραφία [13], είναι τα παρακάτω:

- Ανεμπόδιστη λειτουργία Εισόδου/Εξόδου (**Non-blocking I/O**), με τη χρήση ακολουθούμενης από γεγονότα/συμβάντα (event-driven) αρχιτεκτονικής, όπως παρουσιάζεται στην Εικόνα 2.7, όπου φαίνεται η προσθήκη των event-handlers (και επομένως εργασιών I/O) σε μια ουρά (queue) για εκτέλεση στο παρασκήνιο και επιστροφής

απάντησης όταν αυτή τελειώσει.

- Λειτουργία **single-threat**, για εξοικονόμηση μνήμης RAM, αφού οι multi-threated προσεγγίσεις από άλλους servers (πχ. apache) έδειξαν ότι η δημιουργία ξεχωριστού thread για κάθε session εξαντλούσε τα αποθέματα μνήμης RAM.
- Διαθέτει δικό του διαχειριστή πακέτων, που ονομάζεται NPM (**Node Packet Manger**), για την ευκολότερη εγκατάσταση των υποστηριζόμενων βιβλιοθηκών.
- Μπορεί να εγκατασταθεί σε όλες τις υπάρχουσες πλατφόρμες και λειτουργικά συστήματα (**cross-platform**).



Εικόνα 2. 7: Η επεξεργασία των κλήσεων από το Node.js [15].

Η σημαντική καινοτομία του Node.Js αφορούσε την αξιοποίηση της ήδη γνωστής γλώσσας JavaScript από το επίπεδο του χρήστη (Frontend), με αποτέλεσμα να πάψει η απαίτηση από τους προγραμματιστές να γνωρίζουν μια ξεχωριστή γλώσσα για το επίπεδο του Διακομιστή (backend), όπως συνέβαινε παλαιότερα. Επιπλέον λόγω της single-threated και Non-blocking αρχιτεκτονικής συντελεί στην εξοικονόμηση μνήμης RAM, και συνεπώς στην εξυπηρέτηση πιο απαιτητικών εφαρμογών όπως είναι οι εφαρμογές συνομιλιών πραγματικού χρόνου (Real-Time Communication).

2.10 Nest.JS

Το Nest.JS είναι ένα ισχυρό framework, ανοιχτού κώδικα που αναπτύχθηκε για το επίπεδο του Διακομιστή (Backend) και την ανάπτυξη επεκτάσιμων και συντηρήσιμων εφαρμογών Node.JS. Επιτρέπει στους προγραμματιστές τη δημιουργία δομημένων, συντηρήσιμων, και επεκτάσιμων εφαρμογών διακομιστή αξιοποιώντας σύγχρονες προγραμματιστικές αρχές και τάσεις, όπως τη χρήση της TypeScript.

Τα κύρια χαρακτηριστικά του σύμφωνα με τη βιβλιογραφία του [14], είναι τα παρακάτω:

- Επιδίωξη διαμερισματοποίησης της εφαρμογής σε ξεχωριστές μονάδες (**modules**), καθένα από τα οποία είναι υπεύθυνο για μια συγκεκριμένη λειτουργία ή χαρακτηριστικό της εφαρμογής.

- Αυτόματη φόρτωση-έκχυση των εξαρτήσεων (**dependency injection**), για τη δημιουργία και διαμοιρασμό των στιγμιοτύπων των κλάσεων της εφαρμογής.
- Αξιοποίηση Ελεγκτών (**controllers**) και Δρομολογητών (**routers**), για την επεξεργασία των εισερχόμενων HTTP αιτήσεων (HTTP requests), την κλήση των κατάλληλων μεθόδων που ανήκουν σε συγκεκριμένες υπηρεσίες (services) και επιστροφή απαντήσεων (HTTP responses). Επιπρόσθετα, αξιοποιούν μια σύγχρονη τάση προγραμματισμού, τους διακοσμητές (**decorartors**), οι οποίοι υποδηλώνονται με το σύμβολο "@" το οποίο προηγείται της λέξης-κλειδί που δηλώνει τη σημασιολογία μιας κλάσης (π.χ. @Controller).
- Για την εκτέλεση μεθόδων/συναρτήσεων πριν ή μετά από συγκεκριμένες λειτουργίες (πχ την αποθήκευση ενός document), χρησιμοποιούνται τα **middlewares**[9].
- Φρουρούς **(Guards),** για τη φύλαξη συγκεκριμένων μονοπατιών (paths/routes) με βάση είτε τη πιστοποίηση του χρήστη ή το ρόλο του.
- **Interceptors**, για την τροποποίηση των εισερχόμενων http requests πριν φτάσουν στους controllers και των εξερχόμενων πριν αποσταλούν στο frontend.
- Παρέχεται ένα **CLI (Command Line Interface)** προσφέροντας ένα σύνολο εντολών για άμεση αλληλεπίδραση με το framework.

Παρατηρείται ότι το framework βασίζεται σε δύο παράγοντες: αντλεί έμπνευση από την Angular αξιοποιώντας τη modular αρχιτεκτονική της μαζί με τη δυνατότητα του dependency injection και ενσωματώνει το Express.js για την επεξεργασία των HTTP αιτήσεων. Επιπλέον ενσωματώνει σύγχρονες προγραμματιστικές αρχές και τάσεις, για να αποτελέσει ένα σύγχρονο και περιεκτικό framework για το επίπεδο του Διακομιστή ώστε να παρέχει στους προγραμματιστές να παράγουν επεκτάσιμες, συντηρήσιμες και ευέλικτες εφαρμογές.

2.11 Angular

Το Angular είναι ένα framework ανοιχτού-κώδικα για την ανάπτυξη δυναμικών εφαρμογών διαδικτύου. Έχοντας αναπτυχθεί από την Google [15], η οποία επιπλέον το συντηρεί, επιτρέπει στους προγραμματιστές να αναπτύσσουν εφαρμογές μονής σελίδας (SPAs, Single Page Applications) με δομημένο και οργανωμένο τρόπο. Περιλαμβάνει από προεπιλογή ένα μεγάλο σύνολο από εργαλεία και δυνατότητες για την απλοποίηση της διαδικασίας ανάπτυξης.

Τα βασικά της χαρακτηριστικά, σύμφωνα με τη βιβλιογραφία της [16], είναι τα παρακάτω:

- Βασίζεται στην ανάπτυξη components (**component-based architecture**), τα οποία ενσωματώνουν την όψη ενός HTML element, την αλληλεπίδραση του χρήστη με αυτό, και τη γενική του συμπεριφορά.
- Η όψη ορίζεται σε αρχεία επέκτασης «.html» τα οποία ονομάζονται πρότυπα (**templates**) και μπορεί να περιλαμβάνουν εκτός από απλό HTML κώδικα και καινούριες δυνατότητες της Angular, όπως data-binding[10], directives[11], και event binding[12].
- Τα **directives** αυτά αποτελούν οδηγίες τροποποίησης της δομής ή συμπεριφοράς ενός DOM

[9] https://www.redhat.com/en/topics/middleware
[10] https://angular.io/guide/binding-syntax
[11] https://angular.io/guide/built-in-directives
[12] https://angular.io/guide/event-binding

[17] αντικειμένου, και μερικά προεπιλεγμένα της Angular είναι το «*ngIf», «*ngFor», και «*ngClass». Το σύμβολο αστεριού μπροστά από τα συγκεκριμένα δηλώνει ότι αποτελούν structural directives, που τροποποιούν τη δομή του DOM αντικειμένου.

- Παρέχει τη δυνατότητα έκχυσης αναφορών (**dependency injection**), η οποία παράγει όποτε χρειάζεται εντός της εφαρμογής στιγμιότυπα κλάσεων αυτόματα.
- Η επιχειρηματική λογική συνήθως περιλαμβάνεται στα **services**, στα οποία συναντάμε και την επεξεργασία δεδομένων καθώς και την επικοινωνία με εξωτερικά APIs [18].
- Τα components είναι άρρητα συνδεδεμένα με τα **modules** στα οποία ανήκουν, τα οποία προσφέρουν στην εφαρμογή απομόνωση και αυτονομία των ξεχωριστών λειτουργιών της.
- Η ενσωματωμένη λειτουργία **routing** της Angular, επιτρέπει την πλοήγηση σε διαφορετικές σελίδες της εφαρμογής, χωρίς την απαίτηση επαναφόρτωσης ολόκληρης της web σελίδας.
- Παρέχει τη δυνατότητα δημιουργίας φορμών (**forms**) με τις οποίες αλληλοεπιδρά ο χρήστης, οι οποίες συνοδεύονται και από δυνατότητες ελέγχου ορθότητας (**validation**).
- Δυνατότητα αποστολής HTTP αιτημάτων (HTTP requests) στο επίπεδο διακομιστή, μέσω του **HTTP Client** module που ενσωματώνει.
- Ενσωματώνει τη βιβλιοθήκη **RxJS**, για την υλοποίηση του Event-Listening [19] με **observables** [20] της υπόψη βιβλιοθήκης.
- Διαθέτει τη δυνατότητα εισαγωγής **Pipes**, τα οποία χρησιμοποιούνται στα templates για την τροποποίηση της τιμής πριν αυτή προβληθεί στο χρήστη.
- Παρέχει δυνατότητα **μετάφρασης**, ενσωματώνοντας τη βιβλιοθήκη **“ i18n ”** καθώς και εργαλεία **accessibility** για εξυπηρέτηση ατόμων με ειδικές ανάγκες.
- Και διαθέτει ένα ισχυρό **CLI (Command Line Interface)**, το οποίο διαθέτει ένα σύνολο από εντολές, όπως για τη δημιουργία components/service, και τη χρήση **live development server** στη θύρα 4200.

Με βάση τα παραπάνω, η Angular είναι κατάλληλη για την ανάπτυξη ενός μεγάλου εύρους εφαρμογών, από μικρές προσωπικές, έως μεγάλες εταιρικές, αποτελώντας λόγω των δυνατοτήτων της την κατάλληλη επιλογή για προγραμματιστές που επιθυμούν σύγχρονες, διαδραστικές και συντηρήσιμες διαδικτυακές εφαρμογές.

Κεφάλαιο 3ο: Σχετική Βιβλιογραφία

Στο παρόν κεφάλαιο, γίνεται αναφορά στις διάφορες προσεγγίσεις και των στοιβών ανάπτυξης και τεχνολογιών που επιχειρήθηκαν έως σήμερα για την ανάπτυξη διαδικτυακών εφαρμογών διαχείρισης κρατήσεων και εισιτηρίων και τα αποτελέσματα τους. Εξετάζεται η μέχρι τώρα βιβλιογραφική και ερευνητική πορεία της ανάπτυξης διαδικτυακών εφαρμογών με τη χρήση της στοίβας MEAN. Επιπλέον, το κεφάλαιο αυτό εστιάζει στα κενά που εντοπίζονται σε αυτήν την πορεία, και γίνεται αναφορά στους τρόπους που η έρευνα αυτή απαντά σε αυτά.

3.1 Επισκόπηση των Διαφόρων Προσεγγίσεων

Κατά την ίδρυση του Διαδικτύου, δεν υπήρχαν πολλές επιλογές για την ανάπτυξη κώδικα, πέραν από τη χρήση των βασικών τριών τεχνολογιών **HTML, CSS, JavaScript** για την ανάπτυξη στατικών ιστοσελίδων και της **PHP** για την ανάπτυξη δυναμικών εφαρμογών. Σύμφωνα με τον Tim Berners-Lee [21] η **HTML** (Hypertext Markup Language) αποτελεί μια γλώσσα σήμανσης διαδικτυακού κειμένου. Ένας φυλλομετρητής (browser) διαβάζει τα κείμενα αυτά και τα μετατρέπει σε εικονικές και ακουστικές αναπαραστάσεις χρησιμοποιώντας τις ετικέτες (HTML tags) για να τροποποιήσει το περιεχόμενο τους [22]. Για τον ορισμό κανόνων εμφάνισης (appearance) του διαδικτυακού αυτού κειμένου και της διάταξης του (layout), ο φυλλομετρητής «κατανοεί» τη γλώσσα **CSS** (Cascading Style Sheets) [23]. Η **JavaScript** που αναπτύχθηκε στα πρότυπα του ECMA Script language προτύπου, χρησιμοποιήθηκε αρχικά για την ανάπτυξη script-κώδικα στο επίπεδο του χρήστη (client) ώστε να παρέχει διαδραστικότητα στις διαδικτυακές ιστοσελίδες [23]. Για τη δημιουργία δυναμικών διαδικτυακών σελίδων αναπτύχθηκε μια script-γλώσσα γενικού σκοπού με την ονομασία **PHP** (HyperText Preprocessor), η οποία χρησιμοποιήθηκε για εκτέλεση στο επίπεδο του διακομιστή από ένα PHP processor module. Τέλος, η **MySQL** αναπτύχθηκε ως μια δομημένη γλώσσα επερωτήσεων για σχεσιακές Βάσεις Δεδομένων SQL, αποθηκεύοντας τα δεδομένα της σε δυαδικά (binary) αρχεία.

Χρησιμοποιώντας τις παραπάνω τεχνολογίες και για την ανάπτυξη μιας εξελισσόμενης εφαρμογής κράτησης εισιτηρίων τουρισμού από τους C. Kalu et al [24] χρησιμοποιήθηκε CSS, JS, HTML για το frontend επίπεδο, PHP για το backend επίπεδο και MySQL Βάση Δεδομένων, με αποτελέσματα τα οποία αποδεικνύουν πως ακόμα και με τη χρήση βασικών εργαλείων μπορεί να επιτευχθούν ικανοποιητικά αποτελέσματα εάν αυτά αξιοποιηθούν σωστά. Μία ακόμη προσέγγιση με τα παραπάνω εργαλεία, επιχειρήθηκε από τους Bemile et. al [22] και αφορούσε μια εφαρμογή διαδικτυακής κράτησης εισιτηρίων ξενοδοχείων.

Φτάνοντας στις μέρες μας, έχουν περάσει πολλές τάσεις, κατευθυνόμενες κυρίως από τις επεξεργαστικές ικανότητες και των διαθέσιμων δυνατοτήτων της κάθε εποχής. Στις μέρες μας οι επιλογές αυτές αυξάνονται συνεχώς καθώς οι απαιτήσεις της αγοράς πολλαπλασιάζονται, επιβάλλοντας τη δημιουργία νέων πλαισίων ανάπτυξης (frameworks) που επικεντρώνονται σε συγκεκριμένες απαιτήσεις και όχι την κάλυψη όλου το φάσματος των απαιτήσεων. Μια τέτοια πλατφόρμα ανάπτυξης και εκτέλεσης εφαρμογών είναι το .NET framework, το οποίο ενσωματώνει ένα σύνολο από βιβλιοθήκες, εργαλεία και περιβάλλοντα εκτέλεσης για την ανάπτυξη διαφόρων τύπων εφαρμογών, όπως desktop εφαρμογές (χωρίς τη χρήση του φυλλομετρητή), διαδικτυακές εφαρμογές και υπηρεσίες, υπό την αιγίδα της Microsoft [25]. Η γλώσσα που χρησιμοποιείται ονομάζεται **.NET ή C#**. Αναζητώντας την επαναχρησιμοποίηση της εφαρμογής διαχείρισης κρατήσεων που ανέπτυξαν με το .NET framework, οι Zhou & Chusho [26] διεξήγαγαν έρευνα, από

την οποία συμπέραναν πως οι εφαρμογές κράτησης με μικρότερο εύρος τομέων εξυπηρέτησης (και μεγαλύτερη εξειδίκευση) είχαν μεγαλύτερα ποσοστά επαναχρησιμοποίησης.

Μια άλλη προσέγγιση ανάπτυξης διαδικτυακών εφαρμογών είναι η χρήση της **JSP** (JavaServer Pages) τεχνολογίας, αξιοποιώντας τις δυνατότητες για δυναμικές σελίδες της Servlet κλάσης [27]. Αυτήν την προσέγγιση ακολούθησαν οι Prajapati et. al [28] για την ανάπτυξη εφαρμογής διαχείρισης κρατήσεων αεροπορικών εισιτηρίων, αφού πρώτα πραγματοποίησαν μια συστηματική βιβλιογραφική έρευνα στο πεδίο αυτό. Σε αυτήν αναφέρθηκαν στην σπουδαιότητα των κινητών συσκευών (κυρίως τηλεφώνων), τα οποία αντικατέστησαν και το παραδοσιακό εισιτήριο (χάρτινο), αφού οι ελεγκτές έχουν τη δυνατότητα να «σκανάρουν» τον κωδικό (barcode ή QRcode) από την ίδια την οθόνη του κινητού. Με αυτόν τον τρόπο εξοικονομείται χαρτί συνεισφέροντας στην προστασία του περιβάλλοντος, και ελαχιστοποιούνται οι πιθανότητες να χαθεί το εισιτήριο.

Για την διεξαγωγή ενός ερευνητικού συνεδρίου (conference), και τις απαιτήσεις αποθήκευσης ερευνών/εργασιών σε αποθετήριο, παρουσίασης τους από τους συγγραφείς τους, αξιολόγησης τους από τα μέλη της επιτροπής, και τη δημοσίευση τους επιλέχθηκε από τον Fedaghi [29] να αναπτυχθεί μια υπηρεσία (service), αντί μιας εφαρμογής (application). Μια υπηρεσία (**service**) είναι μια επεξεργαστική αυτόνομη οντότητα, ανεξάρτητη από πλατφόρμες, η οποία αντιθέτως μπορεί να ενσωματωθεί σε μία ή περισσότερες πλατφόρμες. Συνεπώς, η έρευνα του δε στηρίχτηκε σε συγκεκριμένες τεχνολογίες, αλλά αναλύθηκε σε αφαιρετικό επίπεδο, επισημαίνοντας τη σπουδαιότητα της ανεξαρτησίας που παρέχει ένα service, σε αντίθεση με την εξάρτηση ενός application από το φυλλομετρητή (browser) και τις τεχνολογίες που υποστηρίζει.

Ένας άλλος τομέας που επωφελήθηκε σε μεγάλο βαθμό από την εμφάνιση των διαδικτυακών εφαρμογών διαχείρισης κρατήσεων, είναι και αυτός της ιατρικής. Στον τομέα αυτό δεν έχουμε κρατήσεις εισιτηρίων, άλλα διαχείριση ραντεβού εξέτασης. Στον τομέα αυτό αφιέρωσαν την έρευνα τους οι Zhao et al. [30], καταγράφοντας με συστηματικό τρόπο τριάντα έξι (36) άρθρα από την έως εκείνη τη στιγμή βιβλιογραφία. Μερικοί από τους ανασταλτικούς παράγοντες για τη μετάβαση στη χρήση μιας παρόμοιας εφαρμογής που αναδείχθηκαν ήταν το κόστος υλοποίησης της εφαρμογής, η δυνατότητα ελιγμού στο πρόγραμμα, και η έλλειψη διαπροσωπικής σχέσης με τους ασθενείς, εκ των οποίων οι μεγαλύτερες ηλικίες δεν προτιμούν τις τεχνολογικές λύσεις. Παρά τις δυσκολίες αυτές, τα αποτελέσματα τους έδειξαν την εξαιρετικά θετική επίδραση των διαδικτυακών εφαρμογών διαχείρισης των «ραντεβού» (appointments), επιδρώντας στη μείωση των ποσοστών της μη-εμφάνισης σε προγραμματισμένο ραντεβού, στη μείωση των αναγκών σε προσωπικό για τη διαχείριση τους, στη μείωση του χρόνου αναμονής, και την αύξηση της ικανοποίησης των ασθενών.

3.2 Επισκόπηση της Προσέγγισης Εφαρμογών στοίβας MEAN

Στο Κεφάλαιο 3.1 αναφέραμε μερικές προσεγγίσεις ανάπτυξης διαδικτυακών εφαρμογών διαχείρισης κρατήσεων, για να καταλήξουμε σε μία από τις πιο σύγχρονες τάσεις με τη χρήση της στοίβας MEAN, η οποία κάνει χρήση των τεχνολογιών MongoDB, Express.JS/Nest.JS, Angular, Node.JS.

Για την έρευνα της πτυχιακής του, η οποία αφορούσε την ανάπτυξη διαδικτυακής εφαρμογής διαχείρισης κρατήσεων σε καταστήματα sauna, ο Muittari [31] αφού παρουσίασε τις σύγχρονες λύσεις για ανάπτυξη στο επίπεδο του Διακομιστή (backend), επέλεξε να χρησιμοποιήσει τις τεχνολογίες Node.JS και Express.JS,. Για την πιστοποίηση των χρηστών (login), χρησιμοποίησε την

ενσωματωμένη στην Express.JS βιβλιοθήκη Passport, η οποία παρέχει έτοιμες μεθόδους για τον έλεγχο των διαπιστευτηρίων.

Ένα ακόμη είδος διαδεδομένων διαδικτυακών εφαρμογών είναι αυτό των MOOCs (Massive Online Open Courses), οι οποίες προσφέρουν διαδικτυακά μαθήματα υπό μορφή βιντεοσκοπημένων παραδόσεων. Για την ανάπτυξη μιας μινιμαλιστικής έκδοσης MOOC εφαρμογής ο Adhikari [32] χρησιμοποίησε τη στοίβα MEAN, ενσωματώνοντας την MVC αρχιτεκτονική μιας RESTful εφαρμογής.

Ένας τύπος δυναμικών διαδικτυακών εφαρμογών είναι οι CMS (Content Management System), όπως η WordPress[13]. Οι εφαρμογές αυτές επιτρέπουν στους χρήστες μέσα από ένα διαχειριστικό περιβάλλον να επιλέξουν το τελικό αποτέλεσμα μιας εφαρμογής, τόσο ως προς την εμφάνιση, όσο και ως προς τις λειτουργίες της. Για την ανάπτυξη μιας εφαρμογής CMS, ως μέρος της διπλωματικής του ο Thanh [33] χρησιμοποίησε τη στοίβα MEAN, χωρίζοντας την υλοποίηση της σε τρία (3) επίπεδα: μια διαδικτυακή εφαρμογή, μια Εφαρμογή Μονής Σελίδας (SPA, Single Page Application), και τη στοίβα MEAN.

Παρόμοιες έρευνες στον τομέα της ανάπτυξη διαδικτυακών εφαρμογών με τη χρήση της στοίβας MEAN, διενέργησαν και οι Naikwadi et al. [34], Bagal et. al [35], και Nirgudkar et. al [36], για την ανάπτυξη διαδικτυακής εφαρμογής διαχείρισης των στοιχείων φοιτητών πανεπιστημίου, την διερεύνηση της αποτελεσματικότητας στην συνεργασία των τεχνολογιών της στοίβας MEAN, και την ανάλυση καθεμίας τεχνολογίας ξεχωριστά, αντίστοιχα.

Όλες αυτές οι έρευνες αποδεικνύουν την δημοφιλία της στοίβας αυτής στην κοινότητα των προγραμματιστών, η οποία πηγάζει από την αποτελεσματικότητα στην ανάπτυξη απαιτητικών διαδικτυακών εφαρμογών. Μερικοί από τους λόγους που φαίνεται ότι ενισχύουν αυτή τη δημοτικότητα είναι οι παρακάτω:

- Η απαίτηση για συχνές αλλαγές στη δομή της εφαρμογής, που συνεπάγεται αλλαγές και στο σχήμα της Βάσης Δεδομένων,
- Η εξαιρετικά ενεργή κοινότητα προγραμματιστών με ένα εξαιρετικά μεγάλο εύρος βιβλιοθηκών ανοιχτού κώδικα για άμεση χρήση και αποφυγή έναρξης από μηδενική βάση,
- Η απλοποίηση στο συντακτικό του κώδικα που γράφεται και τείνει προς την ανθρώπινη γλώσσα,
- και η πλαισίωση της δομής σε όρια για καλύτερη κατανόηση από νεοεισερχόμενους προγραμματιστές για λόγους συντήρησης και επεκτασιμότητας

Όλοι οι παραπάνω λόγοι συντέλεσαν στην επιλογή της στοίβας MEAN για την ανάπτυξη της διαδικτυακής εφαρμογής διαχείρισης κρατήσεων πολιτιστικών εκδηλώσεων της παρούσας εργασίας.

[13] https://wordpress.com/

Κεφάλαιο 4ο: Μεθοδολογία

Στο παρόν κεφάλαιο περιγράφεται η μεθοδολογία που τηρήθηκε για την ανάπτυξη της εφαρμογής διαχείρισης κρατήσεων για πολιτιστικές εκδηλώσεις, με τη χρήση των τεχνολογιών της στοίβας ανάπτυξης MEAN (MongoDB, Express.JS/Nest.JS, Angular, Node.JS). Η μεθοδολογία αυτή καθορίζει τα βήματα που ακολουθήθηκαν για το σχεδιασμό, την υλοποίηση και την αξιολόγηση της εφαρμογής, καθώς και την προέλευση των δεδομένων.

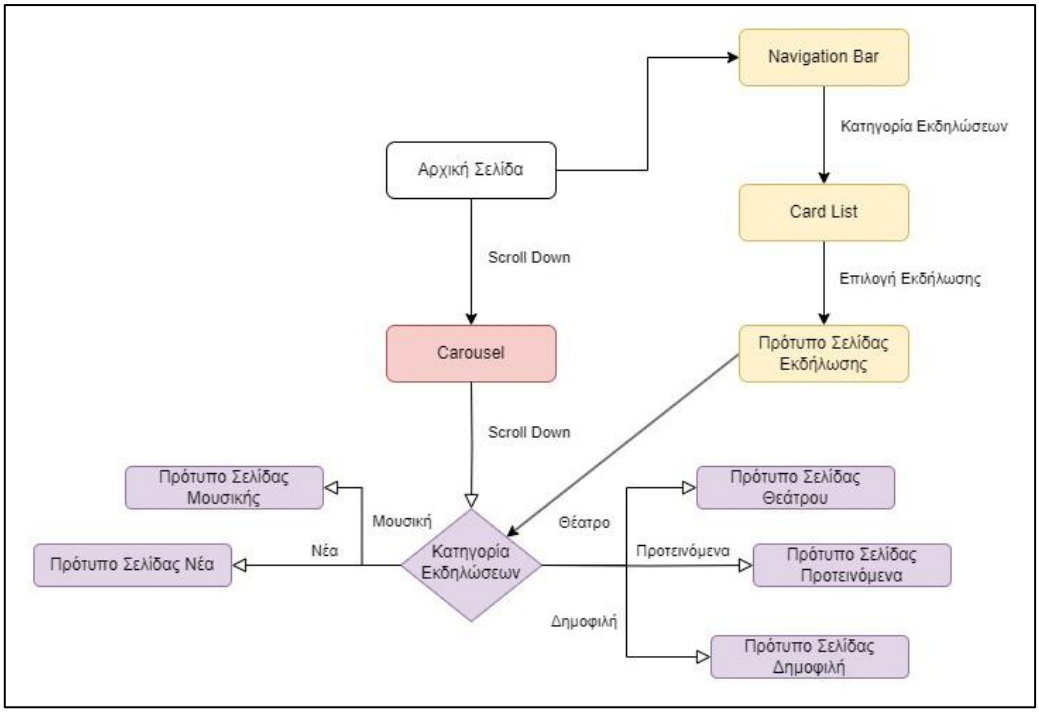
4.1 Σχεδιασμός Αρχιτεκτονικής και Λειτουργικών Απαιτήσεων

Αρχικά πραγματοποιήθηκε ο σχεδιασμός της αρχιτεκτονικής της εφαρμογής. Ορίστηκαν οι βασικοί ρόλοι (διαχειριστές, διοργανωτές, χρήστες) και οι αντίστοιχες Διεπαφές τους (Διαχειριστική Διεπαφή, και Διεπαφή Χρήστη), οι λειτουργικές απαιτήσεις και οι διαφορετικοί τύποι δεδομένων που θα χρειαστεί να αποθηκευτούν.

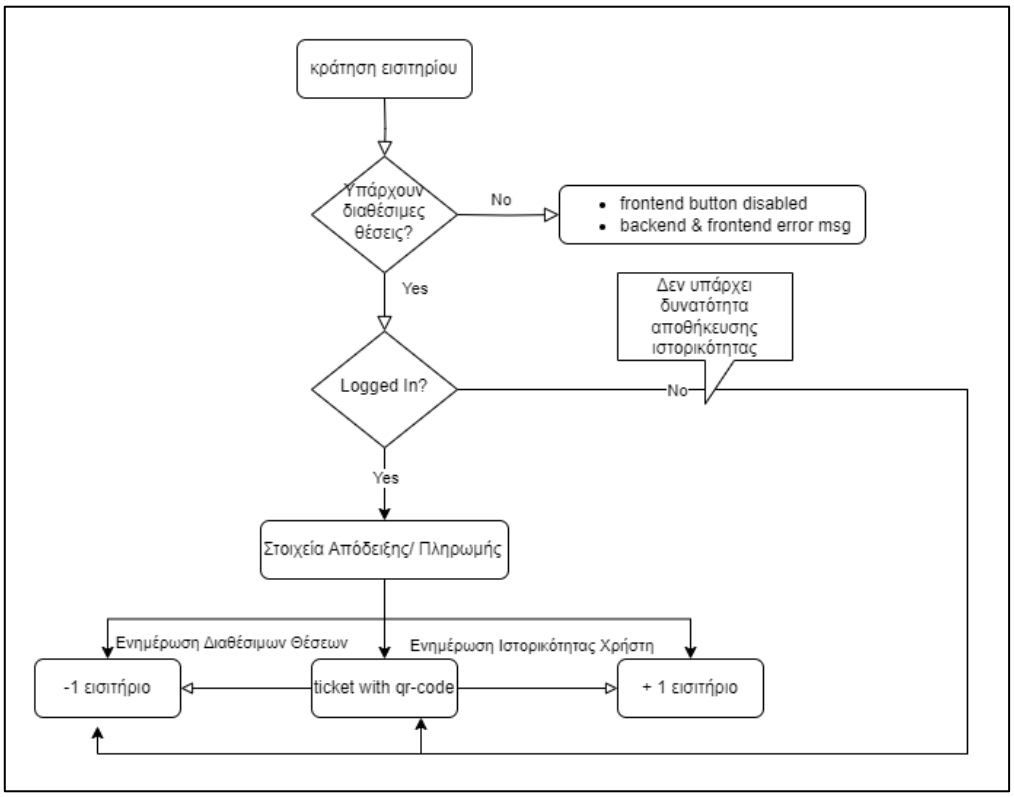
Πιο συγκεκριμένα, οι βασικές λειτουργικές απαιτήσεις από πλευράς **θεατή**, είναι η περιήγηση στις εκδηλώσεις, η κράτηση εισιτηρίου, και η ακύρωση εισιτηρίου. Όπως περιγράφεται και στην Εικόνα 4.1 όπου παρουσιάζεται το διάγραμμα ροής για την περιήγηση στις εκδηλώσεις, ο χρήστης μπορεί να επιλέξει εκδήλωση είτε από τη Μπάρα Πλοήγησης (Navigation Bar) στην κορυφή της Αρχικής Σελίδας, είτε να κυλίσει τη σελίδα προς τα κάτω (scroll down) όπου θα συναντήσει κυλιόμενες κάρτες ανά κατηγορία εκδήλωσης.

- Στην πρώτη περίπτωση, από την Μπάρα Πλοήγησης και επιλέγοντας την κατηγορία της εκδήλωσης ο χρήστης ανακατευθύνεται σε μια σελίδα όπου εμφανίζεται μια λίστα με τις υπάρχουσες εκδηλώσεις της συγκεκριμένης κατηγορίας. Με την επιλογή μιας εκδήλωσης ανακατευθύνεται στη σελίδα της εκδήλωσης, στην οποία περιλαμβάνονται οι σχετικές πληροφορίες της εκδήλωσης, η διαθεσιμότητα των εισιτηρίων και όπου υπάρχει η δυνατότητα κράτησης εισιτηρίου.
- Στη δεύτερη περίπτωση, και αφού ο χρήστης κυλίσει τη σελίδα προς τα κάτω (scroll down), θα μπορεί να επιλέξει από κυλιόμενες κάρτες (carousel) οι οποίες είναι συγκεντρωμένες ανά κατηγορία την εκδήλωση της επιλογής του και να οδηγηθεί επίσης στη σελίδα της εκδήλωσης, στην οποία παρέχονται οι πληροφορίες της εκδήλωσης, η διαθεσιμότητα των εισιτηρίων και υπάρχει δυνατότητα κράτησης εισιτηρίου.

Για την κράτηση εισιτηρίων, όπως παρουσιάζεται στην Εικόνα 4.2, όπου φαίνεται το διάγραμμα ροής της δυνατότητας κράτησης εισιτηρίων, διακρίνουμε δύο περιπτώσεις, η περίπτωση ο χρήστης να είναι συνδεδεμένος (logged in) και αυτή να είναι ως απλός επισκέπτης (guest). Στη πρώτη περίπτωση, τα εισιτήρια που αγοράζονται αποθηκεύονται στο ιστορικο του χρήστη και μπορεί να τα παρακολουθεί, ενώ στη δεύτερη περίπτωση δεν υπάρχει αυτή η δυνατότητα. Επίσης διακρίνουμε ότι στην περίπτωση που δεν υπάρχουν διαθέσιμες θέσεις για την εκδήλωση, ο χρήστης δε μπορεί να επιλέξει να κάνει κράτηση καθώς το κουμπί είναι απενεργοποιημένο.



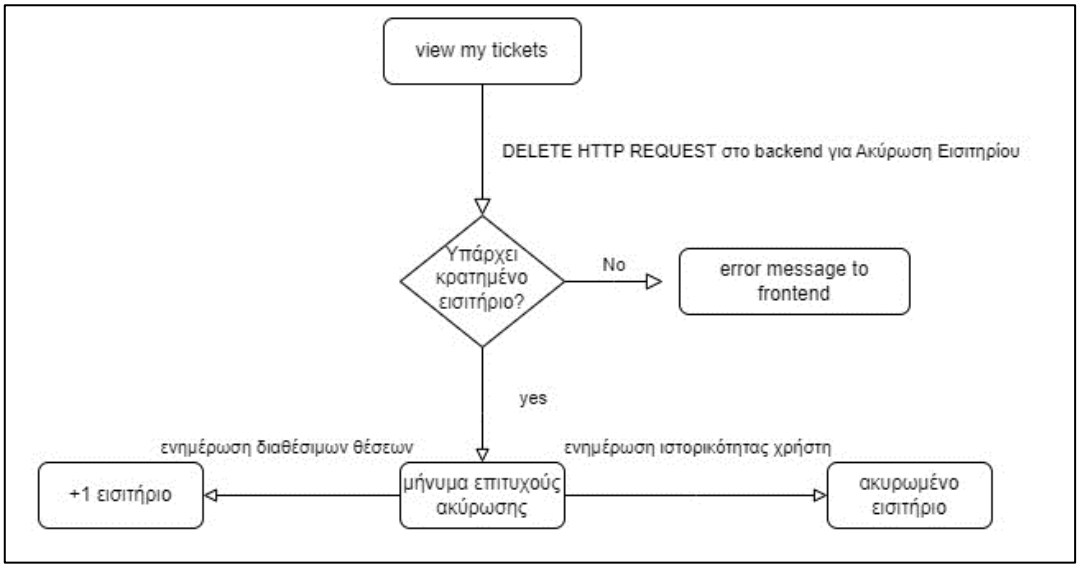
Εικόνα 4. 1: Διάγραμμα Ροής για την Περιήγηση στις Εκδηλώσεις.



Εικόνα 4. 2: Διάγραμμα Ροής για την Κράτηση Εισιτηρίου

Για την ακύρωση εισιτηρίων, και από τη σελίδα ανασκόπησης των κρατημένων εισιτηρίων, θα παρέχεται κουμπί ακύρωσης του εισιτηρίου (μια άλλη δυνατότητα είναι η ακύρωση με τηλεφωνική επικοινωνία, για την οποία θα παρέχεται το τηλέφωνο εξυπηρέτησης). Όπως φαίνεται στην Εικόνα

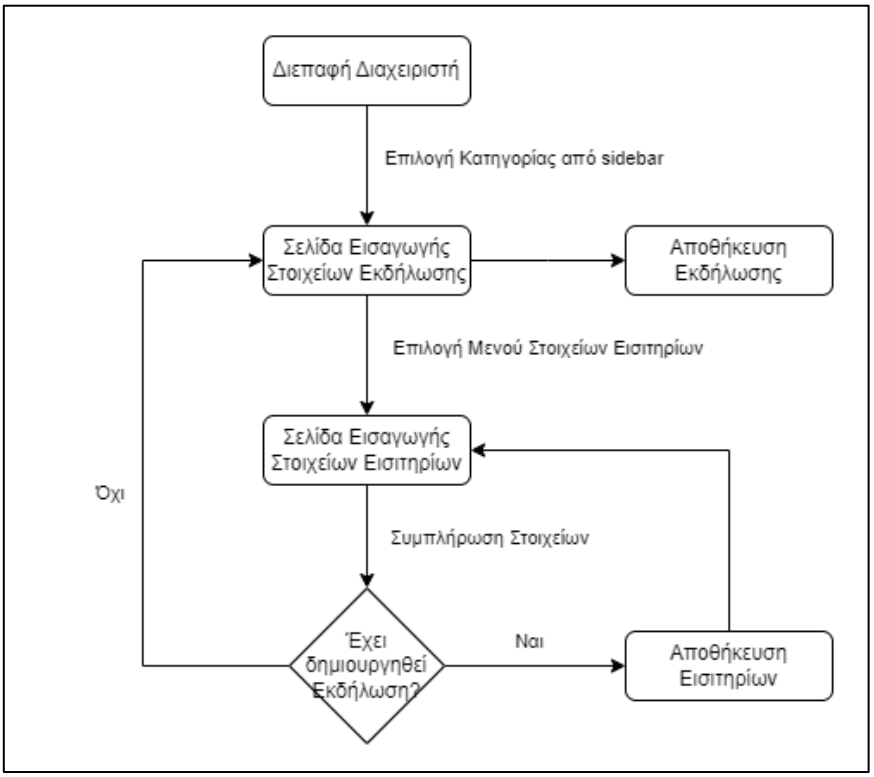
4.3, από τη σελίδα ανασκόπησης και επιλέγοντας την ακύρωση ενός εισιτηρίου, αποστέλλεται ένα DELETE HTTP αίτημα στο διακομιστή για να αιτηθεί την ακύρωση του. Μετά τον έλεγχο της κράτησης, και σε περίπτωση που υπάρχει πραγματοποιείται η ακύρωση και ενημερώνονται η ιστορικότητα του χρήστη και οι διαθέσιμες θέσεις της εκδήλωσης, ενώ σε διαφορετική περίπτωση ενημερώνεται ο χρήστης με ένα μήνυμα σφάλματος.



Εικόνα 4. 3: Διάγραμμα Ροής για την Ακύρωση Εισιτηρίου

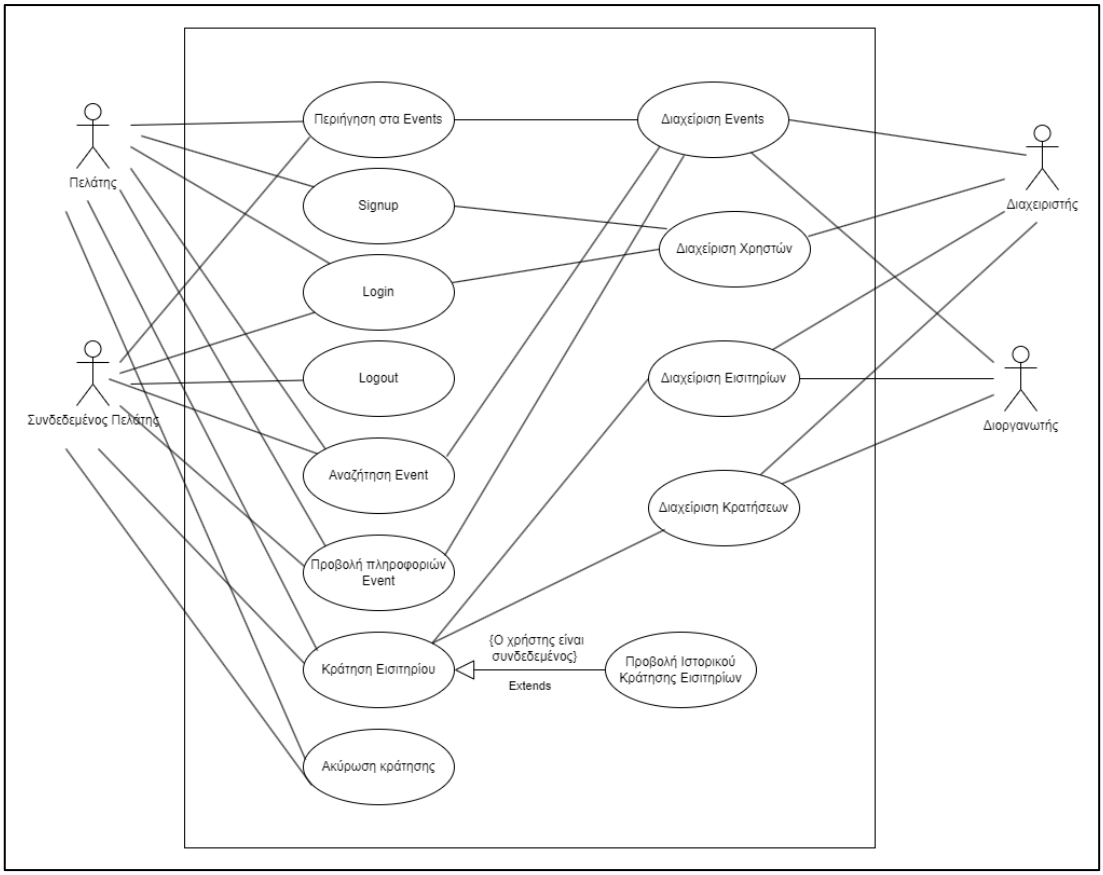
Για τη διεπαφή (UI) του **διοργανωτή/διαχειριστή** συνήθως επιλέγεται ένα περιβάλλον που στοχεύει στην ευκολία διαχείρισης, με απλούστερες φόρμες και λιγότερα γραφικά, όμως μεγάλες ταχύτητες και πολλές δυνατότητες. Για την παρούσα εργασία, οι βασικότερες λειτουργίες που παρέχονται είναι η δημιουργία εκδήλωσης και των εισιτηρίων της, όπως παρουσιάζονται από το Διάγραμμα Ροής της Εικόνας 4.4. Αφού ο χρήστης επιλέξει την κατηγορία της πολιτιστικής εκδήλωσης από το πλαϊνό μενού (π.χ. Θέατρο), ανακατευθύνεται σε μία σελίδα όπου μπορεί να επιλέξει από ένα οριζόντιο μενού τη δημιουργία εκδήλωσης ή τη δημιουργία εισιτηρίων. Αξιοσημείωτο είναι πως προϋπόθεση για τη δημιουργία εισιτηρίων είναι να έχει προηγηθεί η δημιουργία της Εκδήλωσης.

Η παρούσα μεθοδολογία υποστηρίζει την ανάπτυξη μιας ολοκληρωμένης εφαρμογής διαχείρισης κρατήσεων για πολιτιστικές εκδηλώσεις, με έμφαση στην αξιοποίηση των τεχνολογιών της στοίβας ανάπτυξης MEAN (MongoDB, Express.JS/Nest.JS, Angular, Node.JS). Τα προαναφερθέντα διαγράμματα και οι επεξηγήσεις τους, αποτελούν το πλαίσιο εντός του οποίου πραγματοποιήθηκε η ανάπτυξη και η αξιολόγηση της εφαρμογής, εξασφαλίζοντας την ομαλή εξέλιξη της διαδικασίας.



Εικόνα 4. 4: Διάγραμμα Ροής για τη Δημιουργία Εκδήλωσης και των Εισιτηρίων της.

Οι παραπάνω βασικές λειτουργίες της εφαρμογής, συνοψίζονται από τις περιπτώσεις χρήσης που παρουσιάζονται στο διάγραμμα περιπτώσεων χρήσης της Εικόνας 4.5. Σε αυτό διακρίνονται τέσσερις ρόλοι: του Πελάτη, του Συνδεδεμένου Πελάτη, του Διοργανωτή και του Διαχειριστή. Ο διαχειριστής έχει τις δυνατότητες διαχείρισης των εκδηλώσεων, των εισιτηρίων, των χρηστών, και των κρατήσεων τους. Ο Διοργανωτής, έχει τα ίδια δικαιώματα με το Διαχειριστή, πλην της διαχείρισης των χρηστών. Ο χρήστης/πελάτης έχει τις δυνατότητες περιήγησης στις εκδηλώσεις, δημιουργίας λογαριασμού (signup), εισόδου με τα διαπιστευτήρια του (login), εξόδου (logout), αναζήτηση εκδήλωσης από τη μηχανή αναζήτησης, προβολή πληροφοριών εκδήλωσης, των εισιτηρίων τους, καθώς και ακύρωσης κάποιας κράτησης.

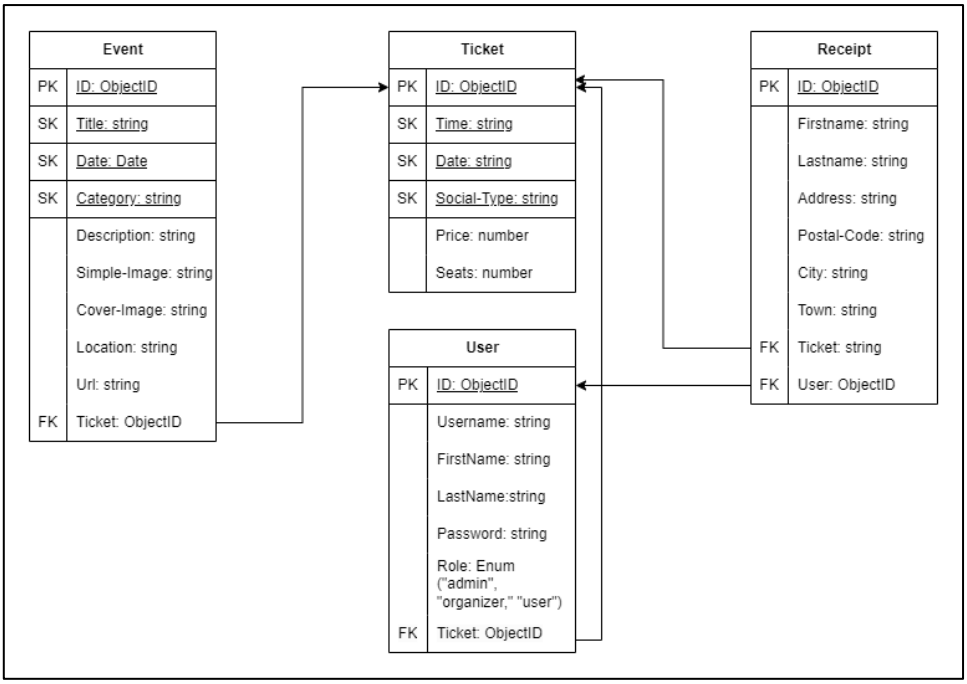


Εικόνα 4. 5: Διάγραμμα Περιπτώσεων Χρήσης της Εφαρμογής.

Έχοντας αναπτύξει θεωρητικά τις λειτουργίες που θα παρέχει η εφαρμογή διαχείρισης κρατήσεων σε πολιτιστικές εκδηλώσεις, διασφαλίζεται η δημιουργία μια λειτουργικής και αποδοτικής πλατφόρμας, και πλαισιώνεται η ανάπτυξη της σε συγκεκριμένα όρια ώστε να αποφευχθούν επαναλήψεις και λάθη που επιφέρουν καθυστέρηση και δυσλειτουργίες.

4.2 Δεδομένα

Σχετικά με τους Τύπους Δεδομένων που χρησιμοποιήθηκαν για τους Τύπους Οντοτήτων της εφαρμογής, αυτοί παρουσιάζονται στην Εικόνα 4.6, όπου παρουσιάζεται το Σχεσιακό Σχήμα της Βάσης Δεδομένων της εφαρμογής διαχείρισης κρατήσεων για πολιτιστικές εκδηλώσεις. Σε αυτό παρουσιάζονται τέσσερις βασικοί Τύποι Οντοτήτων: Η Εκδήλωση, το Εισιτήριο, η Απόδειξη και ο Χρήστης. Ιδιαίτερης σημασίας είναι οι συσχετίσεις μεταξύ των Τύπων Οντοτήτων, οι οποίες προέκυψαν κατόπιν κανονικοποίησης της Βάσης Δεδομένων. Αξιοσημείωτο είναι πως η κανονικοποίηση δεν ήταν αναγκαία, καθώς η MongoDB, μπορεί να χειριστεί και μη Σχεσιακά Δεδομένα ως NoSQL Βάση Δεδομένων. Καθώς όμως η διαχείριση των HTTP αιτημάτων στα πλαίσια μιας RESTful εφαρμογής είναι εξαιρετικά απαιτητική σε NoSQL Βάση Δεδομένων, θεωρήθηκε προτιμότερη η κανονικοποίηση της Βάσης Δεδομένων στα πρότυπα της SQL (Structured Query Language) και η χρήση της ODM βιβλιοθήκης Mongoose.



Εικόνα 4. 6: Σχεσιακό Σχήμα Βάσης Δεδομένων Εφαρμογής

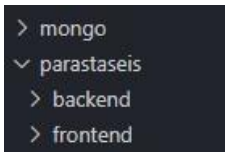
Μια κοινή πρακτική για τη συμπλήρωση των προτύπων εφαρμογών με κείμενο είναι η χρήση του κειμένου «Lorem Ipsum»[14], το οποίο αποτελεί τυχαίο κείμενο και τις περισσότερες φορές είναι αρκετό. Για τις ανάγκες της παρούσας εργασίας, στα πλαίσια επίδειξης (δημιουργίας ενός demo) της εφαρμογής, επιλέχθηκαν τα δεδομένα της Διαδικτυακής Εφαρμογής «more.gr»[15], η οποία αποτέλεσε και στιλιστική πηγή έμπνευσης για το επίπεδο της διεπαφής του χρήστη.

[14] https://www.lipsum.com/
[15] https://www.more.com/gr/

Κεφάλαιο 5ο: Υλοποίηση της Εφαρμογής Διαχείρισης Κρατήσεων

Έχοντας κατανοήσει το θεωρητικό υπόβαθρο των χρησιμοποιούμενων τεχνολογιών στο **Κεφάλαιο 2** και έχοντας καταγράψει τη μεθοδολογία που θα ακολουθηθεί για την ανάπτυξη, θέσαμε την υλοποίηση της εφαρμογής εντός συγκεκριμένων πλαισίων με τα διαγράμματα ροής, περιπτώσεων χρήσης και το Σχεσιακό Σχήμα στο **Κεφάλαιο 3**. Κατόπιν της βιβλιογραφικής επισκόπησης του εξεταζόμενου θέματος της μελέτης στο **Κεφάλαιο 4**, στο παρόν κεφάλαιο παρουσιάζεται η διαδικασία της υλοποίησης μέσα από κομμάτια κώδικα, τόσο στην Angular, όσο και στην Nest.JS. Ολόκληρος ο κώδικας της εφαρμογής, της Angular και του Nest.JS είναι αναρτημένος στο Github αποθετήριο https://github.com/vasoxalipilia/parastaseis. Παρουσιάζεται επίσης, η χρήση του οδηγού (Mongoose) για την εκτέλεση επερωτήσεων στη Βάση Δεδομένων MongoDB.

Για την ευκολότερη εύρεση των αρχείων κώδικα, αυτά ταξινομήθηκαν με την δομή που παρουσιάζεται στην Εικόνα 5.1. Στην ταξινόμηση αυτή, αρχικά έχουμε δύο φακέλους, ο φάκελος «mongo» ο οποίος περιλαμβάνει τα BSON αρχεία της Βάσης Δεδομένων, και ο φάκελος «parastaseis», ο οποίος περιλαμβάνει την εφαρμογή και υποδιαιρείται στον φάκελο «backend» και «frontend». Ο φάκελος backend περιλαμβάνει τον κώδικα του framework «Nest.JS/Express.JS» και ο φάκελος frontend περιλαμβάνει τον κώδικα του framework «Angular».

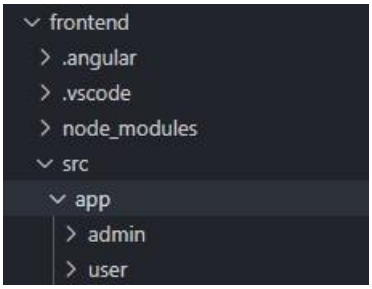


Εικόνα 5. 1: Ταξινόμηση φακέλων κώδικα.

Έχοντας ταξινομήσει τον κώδικα μας με αυτόν τον τρόπο, η αναζήτηση κομματιών κώδικα γίνεται ευκολότερη και η δομή καθίσταται ευανάγνωστη από νεοεισερχόμενους προγραμματιστές που θα κληθούν να συνεχίσουν ή να συνεισφέρουν στην ανάπτυξη του.

5.1 Επισκόπηση της Δυνατότητας «Lazy Loading» της Angular

Επιδιώκοντας την δυνατόν ταχύτερη εξυπηρέτηση του χρήστη στο φυλλομετρητή (browser), η Angular παρέχει τη δυνατότητα για εξεζητημένη φόρτωση κώδικα (lazy loading). Αυτό σημαίνει πως δε θα απαιτηθεί να «φορτωθεί» (download) όλος ο κώδικας της εφαρμογής με τη φόρτωση της σελίδας, αλλά σταδιακά θα κατεβαίνουν τα κομμάτια εκείνα που απαιτούνται όταν επισκεφτεί ο χρήστης τη σελίδα στην οποία ανήκουν. Για να επιτευχθεί αυτό, τα κομμάτια αυτά τα ενσωματώνει ο προγραμματιστής σε συγκεκριμένα «modules», τα οποία καθορίζει να φορτωθούν (download) όταν ο χρήστης επισκεφτεί μια συγκεκριμένα σελίδα (URL/path). Στη παρούσα εφαρμογή αντιστοιχίσαμε τα modules ανάλογα με τις διεπαφές (UI) της, ώστε αν ο χρήστης είναι διαχειριστής και επισκεφθεί τη διαχειριστική σελίδα θα φορτωθεί μόνο το module του διαχειριστή, ενώ εάν είναι απλός χρήστης θα φορτωθεί μόνο το module του χρήστη, εξοικονομώντας εύρος ζώνης (bandwidth), δεδομένα (αν πρόκειται για ογκοχρέωση π.χ. με χρήση κινητού τηλεφώνου) και χρόνο (αφού λιγότερα δεδομένα για φόρτωση σημαίνει ταχύτερη φόρτωση). Όπως παρουσιάζεται στην Εικόνα 5.2, το module του διαχειριστή ονομάστηκε «admin» και των χρηστών «user».

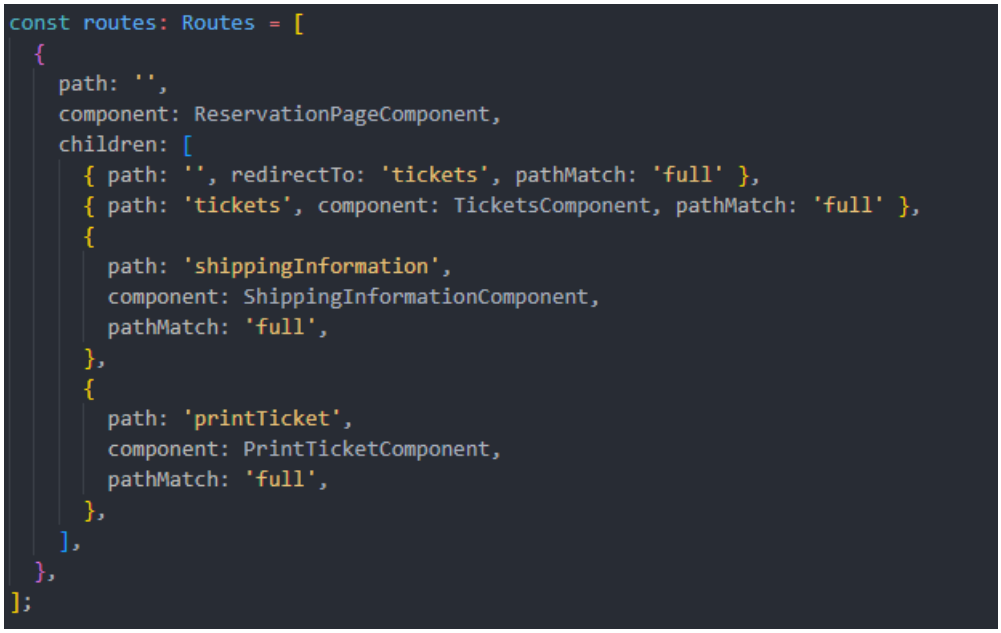


Εικόνα 5. 2: User και Admin Modules της Angular.

Για την υλοποίηση της παραπάνω δυνατότητας, η Angular χρησιμοποιεί συγκεκριμένο συντακτικό, το οποίο παρουσιάζεται στην Εικόνα 5.3. Σε αυτό το συντακτικό, δημιουργείται ένας πίνακας «routes» για όλα τα μονοπάτια (paths, τα οποία αντιστοιχούν σε διευθύνσεις URL) τα οποία μπορεί να επισκεφτεί ο χρήστης. Αυτό δεν αλλάζει και στην περίπτωση που δε χρησιμοποιείται το «lazy loading», όπως παρουσιάζεται στην Εικόνα 5.4 όπου για κάθε path αντιστοιχίζεται ένα «Component». Για κάθε μονοπάτι του «lazy loading» αντιστοιχίζεται πλέον ένα «Module», του οποίου κομμάτια κώδικα θα φορτωθούν στο φυλλομετρητή του χρήστη, όταν αυτός επισκεφθεί το συγκεκριμένο μονοπάτι.

```
const routes: Routes = [
  {
    path: '',
    pathMatch: 'full',
    redirectTo: 'user',
  },
  {
    path: 'user',
    loadChildren: () => import('./user/user.module').then((m) => m.UserModule),
  },
  {
    path: 'admin',
    loadChildren: () =>
      import('./admin/admin.module').then((m) => m.AdminModule),
  },
];
```

Εικόνα 5. 3: Υλοποίηση "lazy loading" στην Angular.



Εικόνα 5. 4: Υλοποίηση "routing" χωρίς "lazy loading" στην Angular.

Με την αξιοποίηση της παραπάνω δυνατότητας, επιτυγχάνεται εξοικονόμηση μεγέθους των αρχείων που απαιτείται να φορτωθούν από το φυλλομετρητή κατά την επίσκεψη της εφαρμογής από το χρήστη. Συνεπώς βελτιώνεται η ταχύτητα, καθώς μειώνεται η αναμονή της φόρτωσης, εξυπηρετώντας συσκευές με χαμηλές δυνατότητες ταχύτητας πλοήγησης στο Διαδίκτυο, όπως τα κινητά τηλέφωνα.

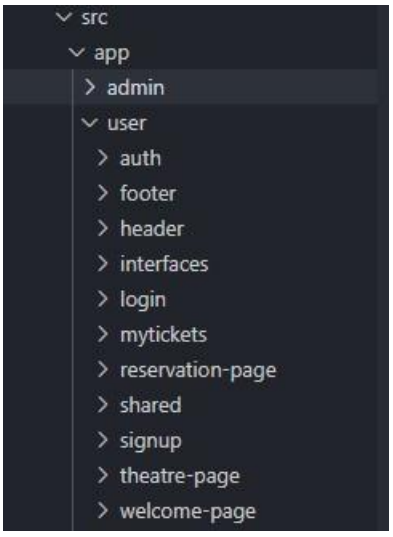
5.2 Επισκόπηση των «Components» της Angular

Κάθε «module» της Angular περιλαμβάνει συγκεκριμένα «Components», για καθένα από τα οποία δημιουργείται ένας ξεχωριστός φάκελος, όπως παρουσιάζεται στην Εικόνα 5.5, στην οποία φαίνονται τα «components» που δημιουργήσαμε για το «module» με την ονομασία «user». Σε αυτό το «Module», περιλαμβάνονται τα παρακάτω:

- **auth** service, το οποίο περιλαμβάνει τον κώδικα με τη μορφή υπηρεσιών (services), για την πιστοποίηση του χρήστη. Συγκεκριμένα υλοποιείται η αποστολή JWT στο διακομιστή,
- **footer** component, το οποίο περιλαμβάνει το component για το υποσέλιδο της εφαρμογής με τα στοιχεία του οργανισμού, καθώς και κάποιους χρήσιμους συνδέσμους,
- **header** component, για την κεφαλίδα των σελίδων της εφαρμογής, η οποία περιλαμβάνει το μενού πλοήγησης για τις διάφορες σελίδες της εφαρμογής,
- **interfaces** κλάση, περιλαμβάνοντας τους τύπους δεδομένων για τον ορισμό των κλάσεων που χρησιμοποιούνται στην εφαρμογή,
- **login** component, το οποίο περιλαμβάνει τον κώδικα για την αυθεντικοποίηση του χρήστη, καλώντας τις μεθόδους της υπηρεσίας της υπηρεσίας auth,
- **mytickets** component, το οποίο περιλαμβάνει την υλοποίηση της επισκόπησης των εισιτηρίων που έχει αγοράσει ο χρήστης,
- **reservation page** component, για την υλοποίηση της κράτησης εισιτηρίων από τους χρήστες. Το παρόν component υποδιαιρείται σε 3 επιπλέον components:

- **tickets**, για την εμφάνιση των πληροφοριών εισιτηρίων της εκδήλωσης,
- **shipping information**, για τη διαχείριση της φόρμας προσωπικών στοιχείων που χρειάζονται για την απόδειξη αγοράς,
- **print tickets**, για την εκτύπωση του εισιτηρίου και τη δυνατότητα αποθήκευσης τοπικά στη συσκευή (πχ υπολογιστής, κινητό).

- **shared** φάκελος, στον οποίο περιλαμβάνονται components τα οποία μπορεί να χρησιμοποιηθούν από διάφορα άλλα components της εφαρμογής, Αποτελείται από τα παρακάτω components:
  - **breadcrumb**, για την υλοποίηση των συνδέσμων του μονοπατιού που ακολουθήθηκε προς την επίσκεψη μιας συγκεκριμένης σελίδας, για την εύκολή πλοήγηση σε μια σελίδα προηγούμενου βήματος,
  - **card-list**, υλοποιώντας τη λίστα καρτών οι οποίες περιλαμβάνουν τις εκάστοτε εκδηλώσεις κάθε κατηγορίας,
  - **carousel**, για την υλοποίηση της παρουσίασης των εκδηλώσεων που παρομοιάζεται με carousel από λούνα-πάρκ με μία εκδήλωση σε κάθε επανάληψη,
  - και **multi-item carousel**, το οποίο είναι παρόμοιο με το παραπάνω carousel, με τη διαφορά ότι περιλαμβάνει τρείς εκδηλώσεις σε κάθε επανάληψη,
- **signup** component, για την υλοποίηση της εγγραφής νέων χρηστών στην εφαρμογή,
- **theatre page** component, το οποίο περιλαμβάνει το πρότυπο σελίδας για εκδηλώσεις που ανήκουν στην κατηγορία θεάτρου. Αντίστοιχα components θα υλοποιηθούν και για τις υπόλοιπες κατηγορίες εκδηλώσεων, καθώς θα διαφέρουν μεταξύ τους.
- **welcome-page** component, περιλαμβάνοντας τον κώδικα για την αρχική σελίδα της διεπαφής του user.

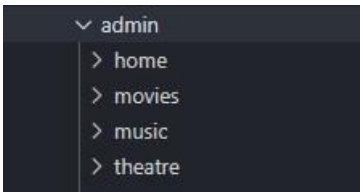


Εικόνα 5. 5: "Components" της διεπαφής/module "user".

Με παρόμοιο τρόπο, έχει αναπτυχθεί και το Module/Διεπαφή «admin» για τους διαχειριστές/διοργανωτές της εφαρμογής, το οποίο όπως φαίνεται στην Εικόνα 5.6 περιλαμβάνει τα

παρακάτω components:

- **home**, το οποίο περιλαμβάνει την αρχική σελίδα της διεπαφής του διαχειριστή, και περιλαμβάνει το καλωσόρισμα και κάποιου είδους πίνακα (dashboard) με ενημερώσεις και χρήσιμα εργαλεία,
- **movies**, για τη διαχείριση της φόρμας εισαγωγής στοιχείων μιας εκδήλωσης και των εισιτηρίων της, για τον λόγο αυτό διακρίνεται περαιτέρω σε δύο επιπλέον components, τα οποία είναι:
    - **event**, για τη φόρμα συμπλήρωσης των στοιχείων της εκδήλωσης,
    - **ticket**, για τη φόρμα συμπλήρωσης των εισιτηρίων της εκάστοτε εκδήλωσης,
- **music**, το οποίο δε διαφέρει σε λειτουργικότητα από το movies component, παρά μόνο στο γεγονός ότι περιλαμβάνει φόρμες για διαφορετική κατηγορία εκδήλωσης. Περιλαμβάνει επίσης αντίστοιχα components: **event** και **ticket**,
- **theatre**, όπως και τα δύο παραπάνω components, με τη διαφορά ότι αφορά συγκεκριμένες φόρμες με πληροφορίες για εκδηλώσεις κατηγορίας θεάτρου, καθώς και για τα εισιτήρια κάθε συγκεκριμένης του εκδήλωσης.



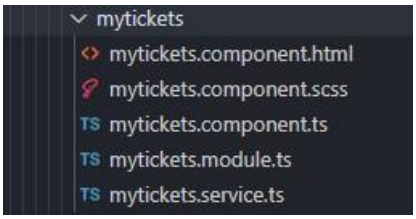
Εικόνα 5. 6: Components της διεπαφής/module "admin".

Με την ταξινόμηση των components της Angular, όπως παρουσιάστηκε στο παρόν κεφάλαιο, διευκολύνεται η κατανόηση από νεοεισερχόμενους προγραμματιστές της δομής της εφαρμογής, και περιορίζεται η πιθανότητα αλλαγής κώδικα σε λανθασμένο αρχείου κατά τη συντήρηση της. Επιπλέον, με τη προσθήκη σε κοινό μέρος (shared φάκελος) των components που επαναχρησιμοποιούνται από πολλαπλά components της εφαρμογής, επιτυγχάνεται και μια άλλη τάση της κοινότητας των προγραμματιστών, η οποία ονομάζεται **DRY** (Don’t Repeat Yourself), και αφορά στην αποφυγή επανάληψης του ίδιου κώδικα σε πολλαπλά σημεία. Η επανάληψη αυτή, επιβάλλει τον έλεγχο των κομματιών κώδικα σε πολλά σημεία, και σε περίπτωση μεταγενέστερης αλλαγής τη διόρθωση σε όλα τα σημεία αυτά, γεγονός που οδηγεί συχνά σε παραλήψεις και σφάλματα.

5.3 Επισκόπηση της MVC Αρχιτεκτονικής στο Frontend της Εφαρμογής

Όπως αναφέρθηκε στο Κεφάλαιο 2.2, η MVC αρχιτεκτονική στοχεύει στο διαχωρισμό της επιχειρηματικής λογικής (business logic) από τη διεπαφή (UI) του χρήστη. Η Angular επένδυσε σε αυτήν την αρχιτεκτονική και όπως φαίνεται στην Εικόνα 5.7, την εισήγαγε στον πυρήνα της ανάπτυξης κώδικα, στα ίδια τα components. Κάθε component, με τη δημιουργία του (μέσω εντολής από το CLI) περιλαμβάνει ένα αρχείο επέκτασης «.component.html», το οποίο αποτελεί την Προβολή (View) της MVC αρχιτεκτονικής και ένα αρχείο επέκτασης «.component.ts», το οποίο αποτελεί το Μοντέλο (Model). «Controller» μπορεί να μη χρειαστούν όλα τα components, άλλα αυτά που το χρειάζονται το λαμβάνουν σε ένα αρχείο επέκτασης «.service.ts». Στο παράδειγμα της εικόνας φαίνονται και δύο άλλα αρχεία, το ένα επέκτασης «.component.scss» το οποίο περιλαμβάνει

κώδικα SCSS[16] (Syntactically Awesome Style Sheet), το οποίο διαμορφώνει στυλιστικά το συγκεκριμένο component, και ένα αρχείο επέκτασης «.module.ts», καθώς το συγκεκριμένο component επιλέχθηκε (για να είναι απολύτως ανεξάρτητο από τα υπόλοιπα components) να περιλαμβάνεται στο δικό του module στο οποίο ανήκει αποκλειστικά μόνο αυτό.

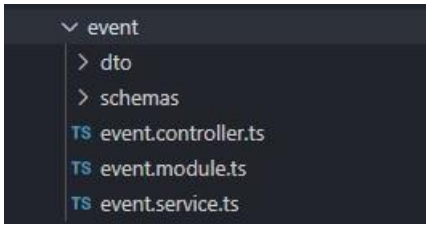


Εικόνα 5. 7: Παρουσίαση αρχείων ενός Component.

Με το παραπάνω παράδειγμα, παρουσιάσαμε τη δομή ενός component, μέσα από τα αρχεία τα οποία περιλαμβάνει. Επιπλέον, αναδείχθηκε η εξάρτηση της Angular από την MVC αρχιτεκτονική, με το διαχωρισμό σε ξεχωριστά αρχεία του κώδικα που αφορά την προβολή (view), το μοντέλο (model) και τον ελεγκτή (controller). Τέλος, αξιοσημείωτο είναι πως κάθε component είναι ανεξάρτητο από τα άλλα, όχι μόνο οργανωτικά σε διαφορετικούς φακέλους, άλλα και ως προς την επικοινωνία μεταξύ τους. Ο κώδικας SCSS/CSS ενός component είναι ανεξάρτητος από τα υπόλοιπα components, συνεπώς αν λάβουμε ως παράδειγμα την ιδιότητα «color:red» μιας κλάσης «container» του component «mytickets», αυτή θα επιβάλει στα γράμματα μόνο της κλάσης «container» του component «myticket» το κόκκινο χρώμα. Οποιοδήποτε άλλο component (πχ. «welcome-page») το οποίο θα περιλαμβάνει και αυτό μια κλάση «container» δε θα επηρεαστεί. Η ίδια ανεξαρτησία και απομόνωση ισχύει και για τον κώδικα Typescript των αρχείων επέκτασης «.component.ts», «.component.html», «.service.ts» και οποιοδήποτε άλλο περιλαμβάνεται σε αυτό.

5.4 Επισκόπηση της MVC Αρχιτεκτονικής στο Backend της Εφαρμογής

Με παρόμοια λογική, η Nest.JS επένδυσε στην MVC αρχιτεκτονική στα πρότυπα της Angular, και όπως φαίνεται στην Εικόνα 5.8, όπου φαίνεται η δομή ενός module με την ονομασία «event», κάθε module αποτελείται από ένα αρχείο επέκτασης «.module.ts» για τον ορισμό του ίδιου του module, ένα αρχείο επέκτασης «.controller.ts» που αντιπροσωπεύει τον Ελεγκτή (Controller) της αρχιτεκτονικής MVC, και ένα αρχείο επέκτασης «.service.ts» που αντιπροσωπεύει το Μοντέλο (Model). Καθώς ο κώδικας του Nest.JS αποτελεί την υλοποίηση του Διακομιστή (Backend) από την Αρχιτεκτονική Τριών Επιπέδων (Three-tier architecture), δεν περιλαμβάνεται σε αυτόν η υλοποίηση της Προβολής (View) από την MVC αρχιτεκτονική, την οποία αναλαμβάνει η Angular.



Εικόνα 5. 8: Παρουσίαση ενός Module/Controller της Nest.JS.

Τέλος, οι controllers των αρχείων επέκτασης «.controller.ts» αναλαμβάνουν να δέχονται HTTP

[16] https://sass-lang.com/

requests, να επεξεργάζονται αυτά τα requests, να επεξεργάζονται τα δεδομένα της εφαρμογής με κλήσεις στη Βάση Δεδομένων μέσω των models (τα οποία έχουν αρχεία επέκτασης «.service.ts») και να επιστρέφουν HTTP responses στην Angular μέσω των ίδιων των controllers.

5.5 Επισκόπηση του Mongoose μέσα από την Nest.JS

Όπως αναφέραμε στο Κεφάλαιο 2.6 η MongoDB είναι μια NoSQL Βάση Δεδομένων, η οποία αποθηκεύει τα δεδομένα σε αρχεία επέκτασης BSON (Binary JSON), τα οποία δε διαθέτουν σχήμα για τη Βάση Δεδομένων. Η Mongoose αποτελεί μία Object Data Modeling (ODM) βιβλιοθήκη σε ένα επίπεδο πάνω από τη MongoDB, εισάγοντας σχήμα και δομή σε μια κατά τα άλλα μη-δομημένη και χωρίς σχήμα Βάση Δεδομένων.

Όταν αξιοποιείται από τη Nest.JS, σύμφωνα με τη βιβλιογραφία [14] ακολουθεί τις παρακάτω αρχές:

- Δομή βασισμένη σε **modules**, ταξινομώντας τις εφαρμογές σε ξεχωριστές μονάδες, περιλαμβάνοντας η κάθε μία ξεχωριστά controllers, services και providers,
- Δυνατότητα **dependency injection** για την αυτόματη δημιουργία στιγμιοτύπων (instances) συγκεκριμένων κλάσεων όταν αυτό απαιτείται.
- Χρήση **Decorators και metadata** για τη δήλωση σημασιολογίας σε κλάσεις, μεθόδους και ιδιότητες. Για το mongoose, τα decorators που απαιτούνται για τη δήλωση των σχημάτων φορτώνονται από το πακέτο “ Nest.JS/mongoose”.
- Διαχείριση Σφαλμάτων (**Error Handling**), με την χρήση φίλτρων (filters) και εξαιρέσεων (exceptions), τα οποία μπορεί να προέρχονται από τη Βάση Δεδομένων.

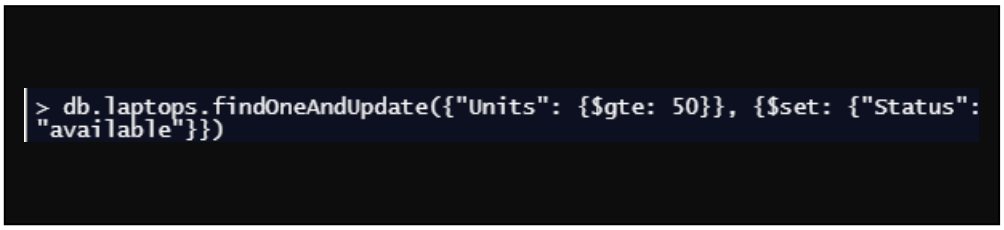
Παράδειγμα χρήσης της mongoose, στην εφαρμογή διαχείρισης κρατήσεων της παρούσας εργασίας, στο επίπεδο του διακομιστή (backend) παρατίθεται στην Εικόνα 5.9. Σε αυτήν παρουσιάζεται μια επερώτηση “findOneAndUpdate” εύρεσης των εισιτηρίων μιας εκδήλωσης (και συγκεκριμένης κοινωνικής κατηγορίας, συγκεκριμένης ημερομηνίας και ώρας) και ενημέρωσης του πεδίου που αναφέρεται στις διαθέσιμες θέσεις (αριθμό διαθέσιμων εισιτηρίων). Στο παράδειγμα αυτό, αφού ενημερωθεί το συγκεκριμένο πεδίο του document Ticket, θα επιστραφεί στο επίπεδο χρήστη (frontend) το ενημερωμένο (με τις ενημερωμένες διαθέσιμες θέσεις) document.

```
async updateTicket(
  event: string,
  socialType: string,
  date: string,
  time: string,
  seats: any,
) {
  return this.ticketModel.findOneAndUpdate(
    { event, socialType, date, time },
    { seats },
    { new: true }, // Return the updated document instead of the old one
  );
}
```

Εικόνα 5. 9: Mongoose μέθοδος για την εκτέλεση επερώτησης findOneAndUpdate.

Σε αντιπαραβολή παρατίθεται στην Εικόνα 5.10 επερώτηση “findOneAndUpdate”, ενός τυχαίου

παραδείγματος στην ίδια τη γλώσσα επερωτήσεων της MongoDB (χωρίς τη χρήση της ODM βιβλιοθήκης Mongoose). Παρατηρούμε ότι και στις δύο περιπτώσεις απαιτείται η χρήση κυρτών αγκυλών σε μορφή JSON, στη μορφή που είναι αποθηκευμένα και τα documents.



Εικόνα 5. 10: MongoDB Επερώτηση findOneAndUpdate .

Με το συνδυασμό της δομημένης σχεδίασης δεδομένων, και το modularity που προσφέρεται από το Nest.JS μαζί με τη δυνατότητα του dependency-injection, οι προγραμματιστές έχουν τη δυνατότητα ανάπτυξης σταθερών και συντηρήσιμων εφαρμογών στο επίπεδο του Διακομιστή (backend) οι οποίες θα συνεργάζονται ομαλά με το επίπεδο του πελάτη (frontend) καθώς και τη Βάση Δεδομένων.

5.6 Επισκόπηση του Nest.JS της Εφαρμογής

Όπως αναφέραμε στο Κεφάλαιο 2.9, το Nest.JS είναι ένα ισχυρό, ανοιχτού κώδικα framework που αναπτύχθηκε για το επίπεδο του Διακομιστή (Backend) και την ανάπτυξη επεκτάσιμων και συντηρήσιμων εφαρμογών Node.JS. Στη εφαρμογή της παρούσας εργασίας, το Nest.JS αξιοποιήθηκε κυρίως για τη διαχείριση των εισερχόμενων HTTP αιτήσεων (HTTP requests) από την Angular (εκτός από την αλληλεπίδραση με τη MongoDB Βάση Δεδομένων που αναλύθηκε παραπάνω).

Όπως φαίνεται στην Εικόνα 5.11, η οποία παρουσιάζει τη μέθοδο “ fetchEventTikets ” (η οποία αναπτύχθηκε για την υποδοχή και απάντηση ενός συγκεκριμένου http request), για τη διαχείριση κάθε τύπου http request (πχ. get, post, put, delete) χρησιμοποιείται το αντίστοιχο decorator συνοδευόμενο από το path/endpoint το οποίο καλεί η αντίστοιχη μέθοδος της Angular. Στο παράδειγμα έχουμε ένα “ @Get “ decorator συνοδευόμενο από το endpoint “ ticket/:event ”, το οποίο σημαίνει πως η συγκεκριμένη μέθοδος θα εκτελεστεί όταν αφιχθεί στην εφαρμογή ένα GET request στο endpoint/path “ ticket/:event “. Επιπλέον στο παράδειγμα αυτό, γίνεται χρήση και μιας παραμέτρου στο endpoint, η οποία ονομάζεται “ event “ και υποδηλώνεται με το σύμβολο “ : “. Παρατηρούμε στη συνέχεια του κώδικα ότι χρησιμοποιείται ένα decorator “ @Param “ το οποίο δέχεται την παράμετρο που στέλνει το επίπεδο χρήστη (Angular), και στη προκειμένη περίπτωση αφορά το id του event (εκδήλωσης) στο οποίο ανήκει το συγκεκριμένο αντικείμενο ticket. Άλλα decorators που χρησιμοποιούνται είναι το “ @Req “ για την υποδοχή του σώματος (Body) του HTTP request, και το “ @Res “ για την δήλωση του http response που θα επιστρέψει η μέθοδος στο επίπεδο χρήστη.

Επιπλέον, παρατηρούμε ότι χρησιμοποιείται η ορολογία **async-await**, η οποία υποδηλώνει ότι πρόκειται για ασύγχρονη μέθοδο [37] και δεν εμποδίζει την εξέλιξη της εφαρμογής άλλα εκτελείται στο παρασκήνιο. Τέλος παρατηρούμε ότι επιστρέφεται στο χρήστη ένα status code 200 (OK), που σημαίνει ότι το request ήταν επιτυχές [38] και το αντικείμενο “ tickets “ αποστέλλεται σε Json μορφή, μορφή «κατανοητή» από το επίπεδο πελάτη (client) και εύκολα μετατρέψιμη σε JavaScript

αντικείμενο (object).

```
@Get('ticket/:event')
async fetchEventTickets(@Req() req, @Res() res, @Param() param) {
  const event = param.event;
  const tickets = await this.eventService.findTicketsByEvent(event);

  return res.status(HttpStatus.OK).json({ tickets });
}
```

Εικόνα 5. 11: Μέθοδος διαχείρισης GET HTTP REQUEST.

Σε γενικές γραμμές, ο Nest.JS κώδικας αναπτύσσεται με δομή, μπορεί να γίνει εύκολα κατανοητός από τρίτους και είναι περιεκτικός, διευκολύνοντας έτσι την ανάπτυξη επεκτάσιμων και συντηρήσιμων εφαρμογών στο επίπεδο Διακομιστή (backend).

Κεφάλαιο 6ο: Παρουσίαση της Εφαρμογής Διαχείρισης Κρατήσεων

Έχοντας καταγράψει τα ζητήματα που σκοπεύει να αντιμετωπίσει η παρούσα εφαρμογή και τα σχεδιαγράμματα πάνω στα οποία κινήθηκε η ανάπτυξη της εφαρμογής διαχείρισης κρατήσεων πολιτιστικών εκδηλώσεων, απομένει η παρουσίαση της διαδικασίας υλοποίησης της. Αυτή στηρίχτηκε στη στοίβα προγραμματισμού MEAN, και στο παρόν κεφάλαιο, αναλύονται οι βασικές λειτουργίες της που στοχεύουν στην κατανόηση της. Επιπλέον παρουσιάζονται στιγμιότυπα (screenshots) από την χρήση της.

Ένας από τους πυλώνες ανάπτυξης της Angular, είναι η «Διαμερισματοποίηση» (modularity), η οποία αναφέρεται στην ανάπτυξη κομματιών κώδικα ως ανεξάρτητα δομικά στοιχεία μιας εφαρμογής, τα οποία μπορούν να σταθούν αυτόνομα χωρίς να εξαρτώνται από άλλα κομμάτια κώδικα (modules), και κυρίως μπορούν να επαναχρησιμοποιηθούν χωρίς να επαναλαμβάνεται ο ίδιος κώδικας. Τα modules αποτελούνται από «Συστατικά» (Components), τα οποία μπορούν με τον ίδιο τρόπο να επαναχρησιμοποιηθούν, αποτρέποντας την επανάληψη των ίδιων κομματιών κώδικα εντός της εφαρμογής.

Η παραπάνω δυνατότητα έδωσε το έναυσμα στην κοινότητα των προγραμματιστών να παράγουν και να διανέμουν (είτε ως κομμάτια ανοιχτού κώδικα, είτε επί πληρωμή) modules και components, για τα οποία η ενσωμάτωση είναι εξαιρετικά απλή και συνήθως συνοδεύεται από οδηγίες από τους δημιουργούς τους.

6.1 Διακόσμηση-Styling της εφαρμογής

Για τη διακόσμηση ιστοσελίδων και Διαδικτυακών εφαρμογών, έχουν αναπτυχθεί πληθώρα βιβλιοθηκών, μερικά παραδείγματα από τις οποίες είναι: Bulma[17], SemanticUI[18],Clarity Design[19], Angular Material[20], και PrimeNG[21]. Όλες παρέχουν ξεχωριστές διαισθητικές υλοποιήσεις, παρέχοντας στους προγραμματιστές τη δυνατότητα επιλογής ανάλογα με τις δικές τους διαισθητικές προτιμήσεις.

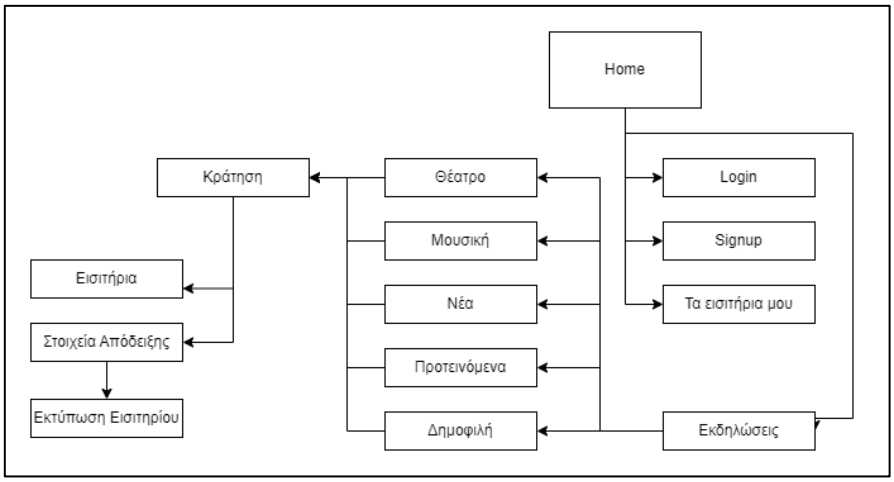
Για τη ανάπτυξη της εφαρμογής Διαχείρισης Κρατήσεων, επιλέχθηκε να μη χρησιμοποιηθούν τα κομμάτια JavaScript κώδικα που παρέχονται από τις παραπάνω βιβλιοθήκες, καθώς η παραμετροποίηση τους είναι ιδιαίτερα απαιτητική. Παρότι, η βιβλιοθήκη «Semantic UI[22]», παρέχει και αυτή τη δυνατότητα ενσωμάτωσης Συστατικών (Components) με ενσωμάτωση κώδικα JavaScript (για τη βιβλιοθήκη React[23]), επιλέχθηκε η εφαρμογή να την αξιοποιήσει μόνο στυλιστικά κάνοντας χρήση του κώδικα CSS (Cascading Style Sheet, [40]). Η λειτουργικότητα των components που προσφέρεται από τη βιβλιοθήκη, όπως η εμφάνιση μια λίστας μενού (dropdown menu), υλοποιήθηκε με την προσθαφαίρεση των απαραίτητων κλάσεων CSS στο αντίστοιχο HTML στοιχείο (element). Κάνοντας χρήση της βιβλιοθήκης αυτής, αξιοποιείται και άλλη απαίτηση των

[17] https://bulma.io/
[18] https://semantic-ui.com/
[19] https://clarity.design/
[20] https://material.angular.io/
[21] https://primeng.org/
[22] https://semantic-ui.com/
[23] https://react.dev/

διαδικτυακών εφαρμογών, αυτή της «προσαρμοστικότητας» (responsiveness) δηλ. να προσαρμόζεται η σελίδα (web page) στις διαστάσεις της οθόνης που χρησιμοποιείται από το χρήστη. Αυτό είναι απαραίτητο στις μέρες μας που το 92.3% των χρηστών του Διαδικτύου χρησιμοποιούν κινητές συσκευές για τη χρήση του [41].

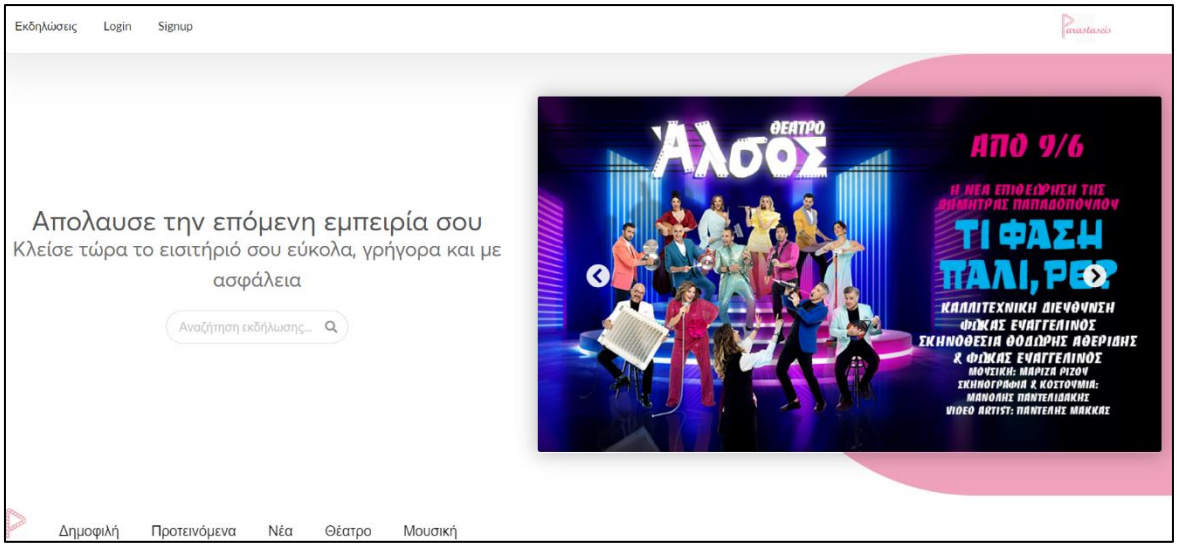
6.2 Παρουσίαση της Εφαρμογής Διαχείρισης Κρατήσεων-Διεπαφή Πελάτη

Η διεπαφή (interface) του **πελάτη**, αποτελείται από δώδεκα σελίδες όπως φαίνεται στην Εικόνα 6.1, όπου παρουσιάζεται ο χάρτης (site-map) της εφαρμογής. Οι κύριες είναι: η Αρχική (Home), Login, Signup, και η «Τα εισιτήρια μου» για την υποδοχή, την είσοδο, την εγγραφή και τη διαχείριση των εισιτηρίων του χρήστη, αντίστοιχα. Στη συνέχεια, από την αρχική, ο χρήστης μπορεί να μεταβεί στις κατηγορίες των εκδηλώσεων από το μενού «Εκδηλώσεις», σε κάθε μία από τις οποίες υπάρχει λίστα καρτών με τις υπάρχουσες και επερχόμενες εκδηλώσεις της υπόψη κατηγορίας. Από την κάθε κατηγορία, ο χρήστης μπορεί να επιλέξει μια εκδήλωση και να ανακατευθυνθεί στη σελίδα της εκδήλωσης, η οποία περιλαμβάνει τις πληροφορίες της εκδήλωσης, και δυνατότητα κράτησης εισιτηρίου. Μετά την επιλογή του εισιτηρίου, παρέχεται η σελίδα συμπλήρωσης των προσωπικών πληροφοριών απόδειξης και τέλος η εκτύπωση του εισιτηρίου.



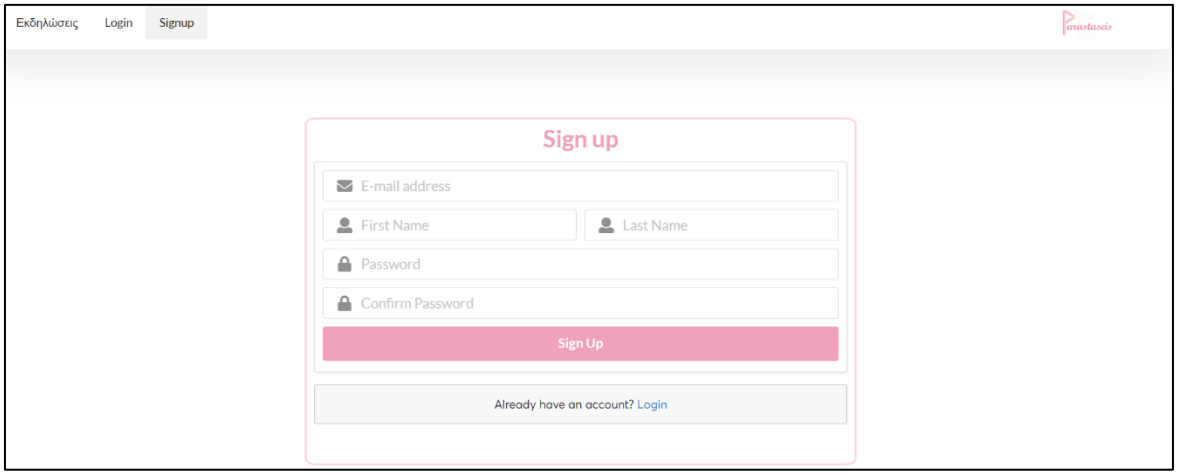
Εικόνα 6. 1: : Site-Map της Διεπαφής Client.

Παρακάτω παρατίθεται μια σύντομη παρουσίαση των σελίδων της εφαρμογής, όπως αυτές ετοιμάστηκαν για την επίδειξη (demo). Αρχικά, ο χρήστης επισκεπτόμενος την εφαρμογή συναντάει την Αρχική σελίδα (Home), στην οποία όπως φαίνεται στην Εικόνα 6.2, υπάρχει ένα μενού πλοήγησης (Navigation Bar), όπου ο χρήστης μπορεί να επιλέξει τις κατηγορίες των εκδηλώσεων, να πραγματοποιήσει είσοδο (login) στην εφαρμογή, είτε να εγγραφεί (signup). Στην περίπτωση που ο χρήστης είναι συνδεδεμένος, αντί των «login» και «Signup» εμφανίζεται το κουμπί «Logout». Επιπλέον, εμφανίζεται για διαισθητικούς λόγους και ένα carousel από φωτογραφίες με διαφημιζόμενες εκδηλώσεις, καθώς και μια μηχανή αναζήτησης για εύκολη και άμεση αναζήτηση εκδηλώσεων με βάση λέξεις-κλειδιά.



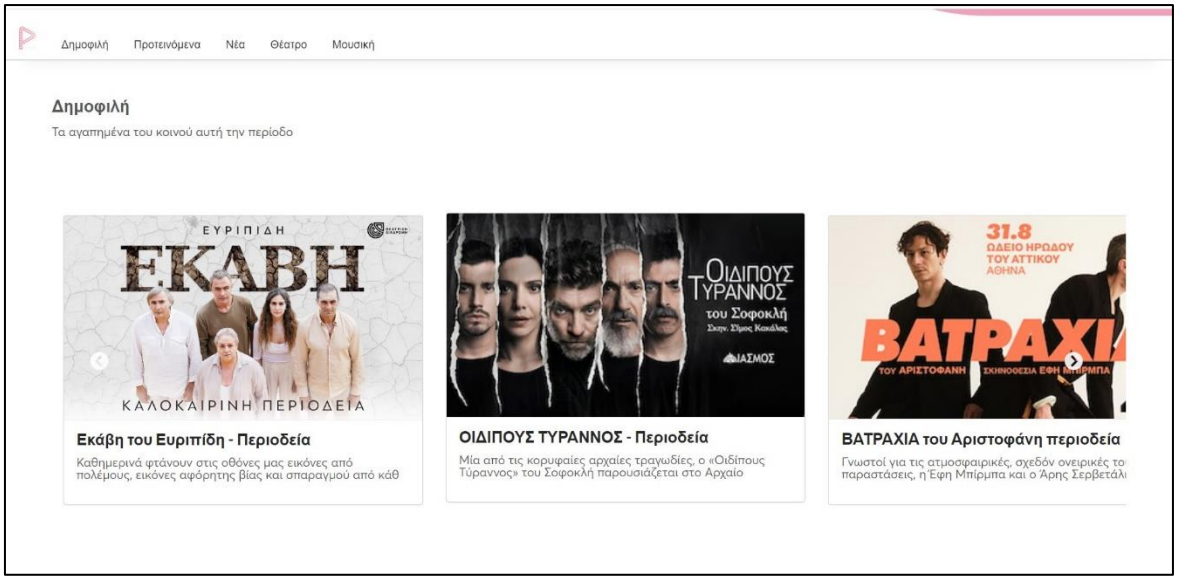
Εικόνα 6. 2: Αρχική Σελίδα εφαρμογής.

Η σελίδα με τη φόρμα εγγραφής (Signup) παρουσιάζεται στο στιγμιότυπο της Εικόνας 6.3, όπου ο χρήστης παροτρύνεται να εισάγει τη Διεύθυνση Ηλεκτρονικού Ταχυδρομείο (E-mail Address) ως Username, το Όνομα και το Επίθετο του και το συνθηματικό κλειδί (με την επιβεβαίωση του για αποφυγή τυπογραφικών λαθών).



Εικόνα 6. 3: Σελίδα Signup.

Πραγματοποιώντας κύλιση (scroll down) της σελίδας, και κατεβαίνοντας προς το χαμηλότερο σημείο της αρχικής σελίδας, το μενού πλοήγησης (navigation bar) αντικαθίσταται από ένα νέο, το οποίο περιλαμβάνει μόνο τις κατηγορίες των εκδηλώσεων. Επιπλέον, όπως φαίνεται στο στιγμιότυπο της Εικόνας 6.4, για κάθε κατηγορία υπάρχει ένα carousel (με πολλαπλά στοιχεία, έναντι του ενός που υπάρχει στη σελίδα παραπάνω) με τις εκδηλώσεις της ως υπερσύνδεσμοι, από τις οποίες ανακατευθυνόμαστε στην σελίδα της εκάστοτε εκδήλωσης. Κάθε κάρτα εκδήλωσης περιλαμβάνει τον τίτλο, μια φωτογραφία, και ένα απόκομμα της περιγραφής.

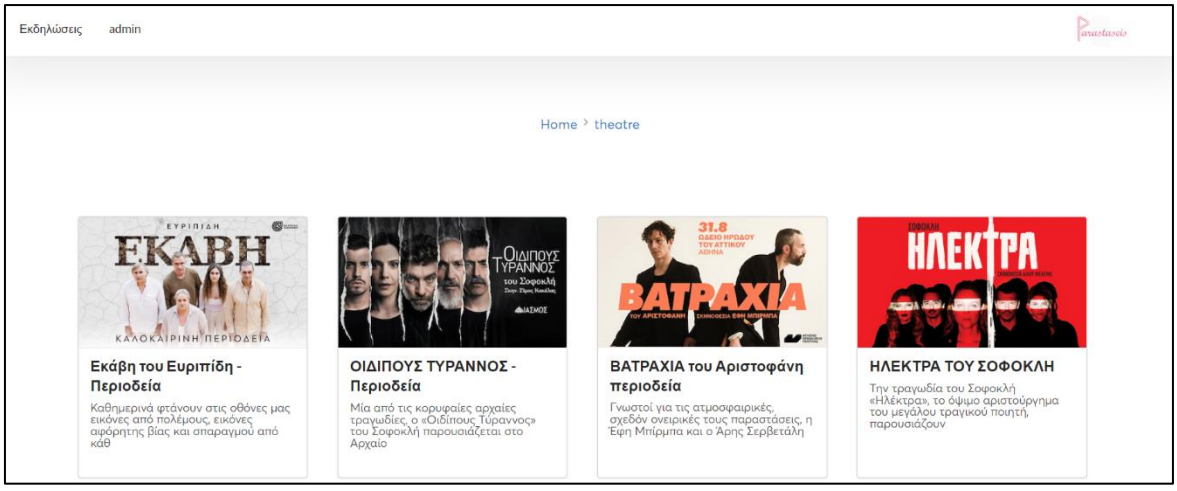


Εικόνα 6. 4: Αρχική σελίδα (έπειτα από scroll down).

Μεταβαίνοντας με κύλιση προς την κάτω κατεύθυνση, και αφού έχουν εμφανιστεί όλες οι κατηγορίες των εκδηλώσεων, στο τέλος της σελίδας βρίσκεται το υποσέλιδο (footer), το οποίο όπως φαίνεται στην Εικόνα 6.5 περιλαμβάνει πληροφορίες σχετικά με τον οργανισμό, τα στοιχεία επικοινωνίας, και άλλες παρόμοιες πληροφορίες.

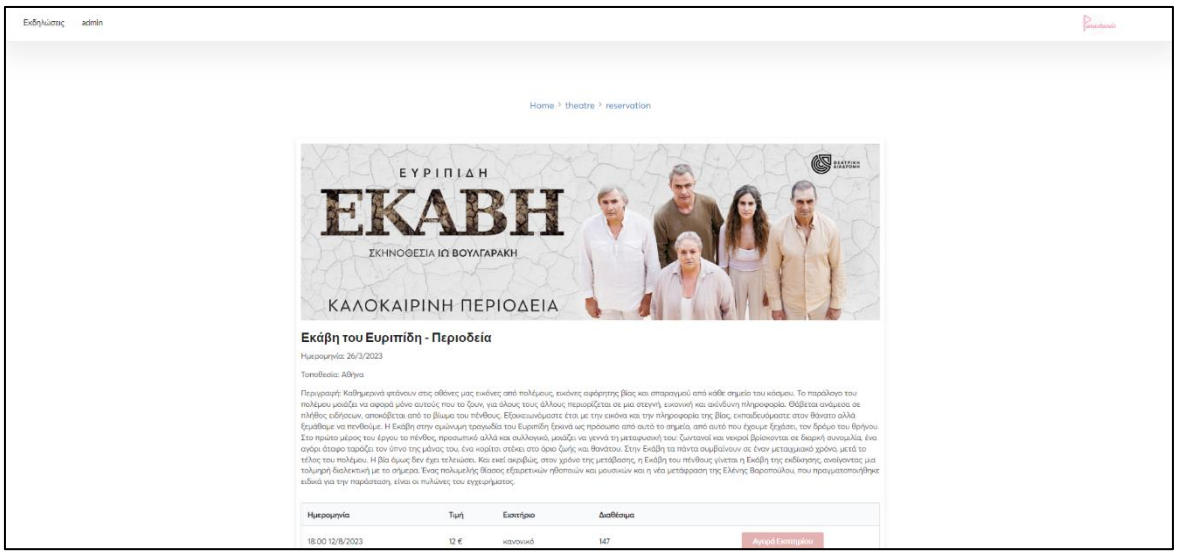
Εικόνα 6. 5: Υποσέλιδο (footer) εφαρμογής.

Επιλέγοντας κάποιον σύνδεσμο από τις διαθέσιμες κατηγορίες που βρίσκονται στο αρχικό μενού πλοήγησης, ο επισκέπτης ανακατευθύνεται στη σελίδα της συγκεκριμένης κατηγορίας, όπου εμφανίζεται μια λίστα με τις διαθέσιμες εκδηλώσεις της κατηγορίας, όπως παρουσιάζεται στο στιγμιότυπο της Εικόνας 6.6, όπου έχει επιλεγεί η κατηγορία «θέατρο».



Εικόνα 6. 6: Στιγμιότυπο από τη σελίδα της κατηγορίας "θέατρο".

Αφού ο χρήστης κάνει «κλικ» στην εκδήλωση της επιλογής του, ανακατευθύνεται στη σελίδα της εκάστοτε εκδήλωσης όπως στην Εικόνα 6.7, για την οποία περιλαμβάνονται πληροφορίες, όπως ο τίτλος, η τοποθεσία, η περιγραφή της εκδήλωσης, και τα διαθέσιμα εισιτήρια, ταξινομημένα κατά ημερομηνία προβολής. Επιπλέον, περιλαμβάνεται η τιμή, του εκάστοτε εισιτηρίου, ο τύπος (δλδ. εκπτωτικό/κανονικό), οι διαθέσιμες θέσεις, και ένα κουμπί για την αγορά του εισιτηρίου, το οποίο σε περίπτωση που δεν υπάρχουν διαθέσιμες θέσεις, θα γίνει απενεργοποιημένο (disabled).

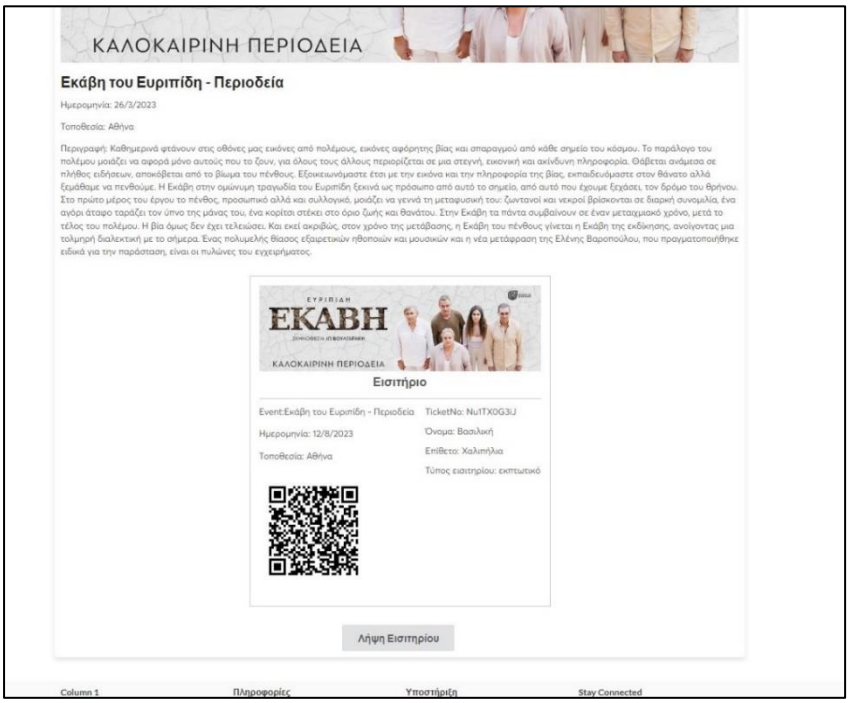


Εικόνα 6. 7: Σελίδα Εκδήλωσης.

Επιλέγοντας το κουμπί αγοράς, τα διαθέσιμα εισιτήρια αντικαθίσταται από μία φόρμα εισαγωγής προσωπικών στοιχείων για την απόδοση απόδειξης, όπως φαίνεται στην Εικόνα 6.8. Υπάρχει η δυνατότητα επιλογής του κουμπιού «προηγούμενο» για να εμφανιστεί πάλι η λίστα με τα διαθέσιμα εισιτήρια ώστε να επιλεγεί κάποιο άλλο εισιτήριο, είτε «αγορά» για τη οριστικοποίηση της αγοράς και τη μετάβαση στη σελίδα εκτύπωσης του εισιτηρίου.

Εικόνα 6. 8: Σελίδα εισαγωγής προσωπικών στοιχείων απόδειξης.

Μετά τη συμπλήρωση των προσωπικών στοιχείων και την επιλογή του κουμπιού αγορά, η φόρμα αντικαθίσταται από μια μικρογραφία του εισιτηρίου, και ένα κουμπί για την αποθήκευση του τοπικά στον υπολογιστή του χρήστη ως αρχείο «.pdf», όπως φαίνεται στο στιγμιότυπο της Εικόνας 6.9.



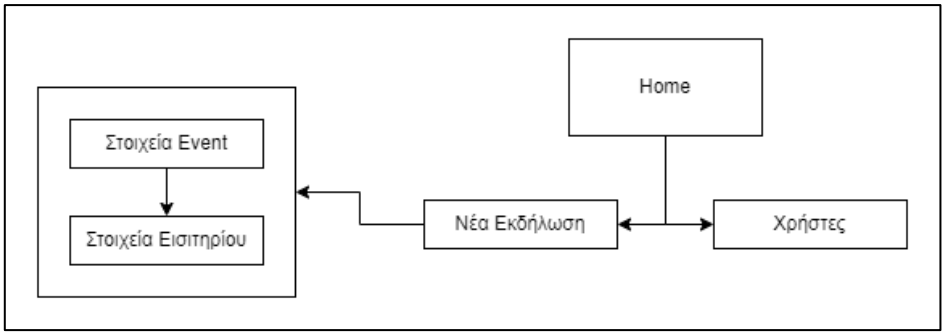
Εικόνα 6. 9: Σελίδα Λήψης Εισιτηρίου.

Αξιοσημείωτο είναι πως ο μοναδικός κωδικός του εισιτηρίου, καθώς και το ονοματεπώνυμο του πελάτη αποτυπώνονται σε «QR-code» που θα εκτυπωθεί στο εισιτήριο για τον γρήγορο έλεγχο από τους υπεύθυνους ασφαλείας της εκδήλωσης.

6.3 Παρουσίαση της Εφαρμογής Διαχείρισης Κρατήσεων-Διεπαφή Διαχειριστή.

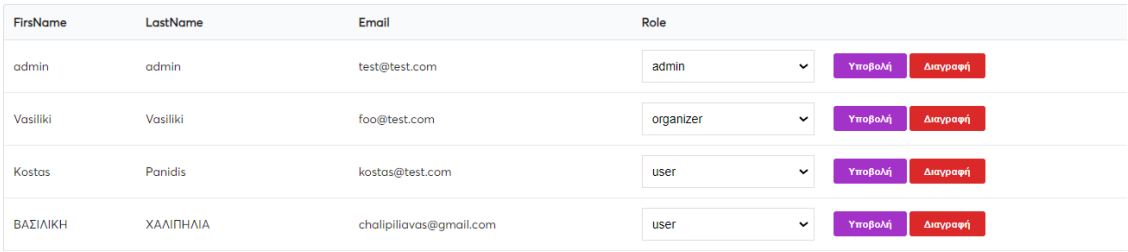
Η διεπαφή (interface) του **διαχειριστή**, αποτελείται από πέντε σελίδες, οι οποίες παρουσιάζονται σε διάγραμμα χάρτη (site-map) στην Εικόνα 6.10. Οι κύριες είναι οι «Home», «Χρήστες», και «Νέα εκδήλωση», για την υποδοχή του χρήστη, τη διαχείριση των χρηστών της εφαρμογής και τη

δημιουργία νέας εκδήλωσης, αντίστοιχα. Στη διεπαφή αυτή, έχουν δικαίωμα να εισέλθουν μόνο όσοι χρήστες έχουν ρόλο «Διαχειριστή».



Εικόνα 6. 10: Site-Map της Διεπαφής Admin.

Προσθέτοντας την λέξη «admin» στο φυλλομετρητή, έπειτα από το URL[24] (Uniform Resource Locator) της εφαρμογής (δηλ. <URL>/admin), εμφανίζεται η Διεπαφή του διαχειριστή, εάν έχει παραχωρηθεί ο ρόλος του Διαχειριστή ή Διοργανωτή (admin/organizer), σε διαφορετική περίπτωση του απαγορεύεται η πρόσβαση και ανακατευθύνεται στην αρχική σελίδα αναζήτησης εκδηλώσεων. Από το οριζόντιο αριστερό μενού (sidebar), είναι δυνατή η διαχείριση των χρηστών της εφαρμογής με την επιλογή του μενού «Χρήστες», και συγκεκριμένα η αλλαγή το ρόλου ενός χρήστη σε μια από τις τρείς δυνατές επιλογές (admin, organizer, user). Ένα στιγμιότυπο της σελίδας εμφανίζεται στην Εικόνα 6.11, όπου φαίνεται η επιλογή ρόλων μέσω του dropdown list, η υποβολή μέσω του κουμπιού «Υποβολή» καθώς και η διαγραφή .



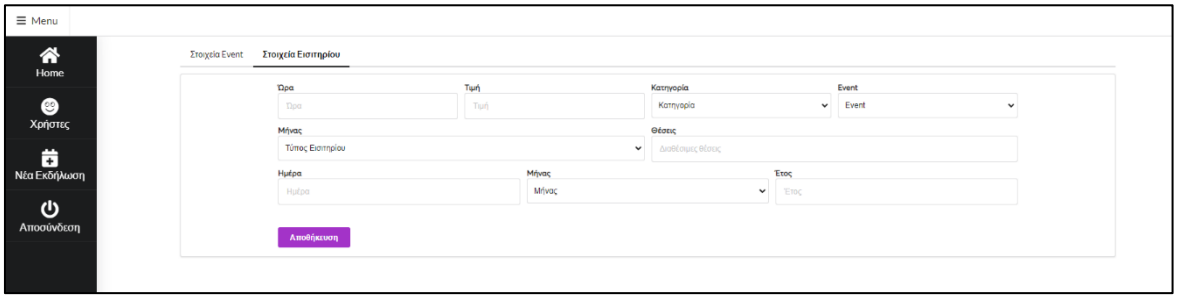
Εικόνα 6. 11: Στιγμιότυπο διαχειριστικής σελίδας διαχείρισης χρηστών.

Ακόμη είναι δυνατή η δημιουργία μιας εκδήλωσης (π.χ. θέατρο), με την επιλογή του κουμπιού «Νέα εκδήλωση» από το οριζόντιο μενού (sidebar). Μετά την επιλογή, εμφανίζεται η σελίδα συμπλήρωσης στοιχείων της εκδήλωσης, όπως φαίνεται στο στιγμιότυπο της Εικόνας 6.11. Η σελίδα αυτή διαθέτει δύο κουμπιά για τη μεταφόρτωση (upload) εικόνας για κάθε εκδήλωση. Μια εικόνα με διαστάσεις τετραγώνου σχήματος, ώστε να προσαρμόζεται στις κάρτες των carousel, και μια εικόνα με διαστάσεις παραλληλογράμμου σχήματος ώστε να τοποθετηθεί ως κεφαλίδα (header) στη σελίδα της εκδήλωσης (και οι δύο αναφέρονται στη διεπαφή του χρήστη). Μετά τη συμπλήρωση των στοιχείων επιλέγεται η αποθήκευση.

[24] https://en.wikipedia.org/wiki/URL

Εικόνα 6. 12: Σελίδα εισαγωγής στοιχείων εκδήλωσης - Διεπαφή Διαχειριστή.

Στη συνέχεια, και αφού έχει δημιουργηθεί πρώτα η εκδήλωση, είναι δυνατή η επιλογή από το οριζόντιο μενού του συνδέσμου «Στοιχεία Εισιτηρίων», όπου συμπληρώνονται τα στοιχεία των εισιτηρίων για κάθε εκδήλωση, όπως φαίνεται στο στιγμιότυπο της Εικόνας 6.12.



Εικόνα 6. 13: Σελίδα εισαγωγής στοιχείων εισιτηρίων εκδήλωσης - Διεπαφή Διαχειριστή.

Η Διεπαφή του διαχειριστή συνηθίζεται να περιλαμβάνει κυρίως φόρμες, και εργαλεία διαχείρισης, χωρίς ιδιαίτερα γραφικά και να στοχεύει στην πρακτικότητα, στην ταχύτητα και στη σταθερότητα. Η Διεπαφή του διαχειριστή της παρούσας εφαρμογής δεν αποτελεί εξαίρεση, επικεντρώνοντας στην αποτελεσματική διαχείρισης των εκδηλώσεων για τους διοργανωτές.

Κεφάλαιο 7ο: Ανασκόπηση της εργασίας - Συμπεράσματα

Στα προηγούμενα κεφάλαια διαπραγματευτήκαμε την ανάπτυξη μιας εφαρμογής διαχείρισης κρατήσεων σε πολιτιστικές εκδηλώσεις, με την αξιοποίηση της στοίβας προγραμματισμού MEAN, και συγκεκριμένα τα πακέτα ανάπτυξης (frameworks) Angular, Nest.JS /Express.JS, Node.JS, και MongoDB. Επιπλέον, αναπτύχθηκε το θεωρητικό υπόβαθρο των χρησιμοποιούμενων τεχνολογιών, για την ευκολότερη κατανόηση της διαδικασίας που τηρήθηκε. Το παρόν κεφάλαιο παρέχει μια επισκόπηση της παρούσας εργασίας και ανοίγει το δρόμο για την μελλοντική ανάπτυξη και έρευνα.

7.1 Ανασκόπηση

Αρχικά, στο **Κεφάλαιο 2**, αναλύθηκαν οι βασικές έννοιες και αρχές πάνω στις οποίες στηρίχτηκε η ανάπτυξη της εφαρμογής Διαχείρισης Κρατήσεων Πολιτιστικών Εκδηλώσεων. Πιο συγκεκριμένα, παρουσιάστηκε η Αρχιτεκτονική πολλαπλών επιπέδων (multi-tier architecture) με τρία κύρια επίπεδα. Αυτά είναι το επίπεδο Πελάτη (frontend), τον ρόλο του οποίου αναλαμβάνει η Angular, το επίπεδο Διακομιστή (backend) που αναλαμβάνει η Nest.JS/Express.JS, και το επίπεδο Βάσης Δεδομένων (database), στο οποίο επιλέχθηκε η MongoDB. Επιπλέον, στο Κεφάλαιο αυτό αναφέρεται η MVC (Model, View, Controller) αρχιτεκτονική και η ενσωμάτωση της στα frameworks Angular, και Nest.JS, καθώς και διάφορες λύσεις πιστοποίησης χρηστών ανοιχτού κώδικα που χρησιμοποιούνται στον επιχειρηματικό κλάδο, ώστε να καταλήξουμε στην ιδανική λύση για την παρούσα εργασία, την τεχνολογία JWT (Json Web Token). Η τελευταία χρησιμοποιεί ισχυρή κρυπτογραφία (Ασσύμετρη Κρυπτογραφία Δημόσιου/Ιδιωτικού κλειδιού) για την ασφαλή μεταφορά μηνυμάτων μεταξύ πελάτη (frontend/client) και διακομιστή (backend/server) και για την πιστοποίηση των χρηστών της εφαρμογής. Τέλος, αναλύθηκε η τάση των προγραμματιστών για RESTful (Representation State Transfer) εφαρμογές, για να επιτευχθεί ευελιξία στην ανάπτυξη, απομόνωση των μεθόδων και απλότητα στην κατανόηση των αλληλεπιδράσεων των πόρων της εφαρμογής.

Στο **Κεφάλαιο 3**, εξετάστηκε η μέχρι τώρα βιβλιογραφική και ερευνητική πορεία της ανάπτυξης διαδικτυακών εφαρμογών με τη χρήση τόσο διαφορετικών τεχνολογιών από αυτά της παρούσης εργασίας, όσο και με τη χρήση της στοίβας ανάπτυξης MEAN.

Το **Κεφάλαιο 4** επικεντρώθηκε στη παρουσίαση της Μεθοδολογίας που χρησιμοποιήθηκε κατά την ανάπτυξη της εφαρμογής Διαχείρισης Κρατήσεων, αφού έχουν προηγουμένως έχουν γίνει οικεία τα αντικείμενα που διαπραγματεύεται. Αυτή καθορίζει τα βήματα που ακολουθήθηκαν για το σχεδιασμό, την υλοποίηση, και την αξιολόγηση της εφαρμογής. Επιπλέον, έγινε αναφορά στην πηγή στυλιστικής έμπνευσης, αλλά και των δεδομένων, ώστε το αποτέλεσμα της επίδειξης (demo) να αποκτήσει ενδιαφέρον. Για την επίτευξη των παραπάνω, αξιοποιήθηκαν διαγράμματα διαφόρων ειδών, όπως Περιπτώσεων Χρήσης, Ροής, και το Σχεσιακό Σχήμα της Βάσης Δεδομένων.

Στο **Κεφάλαιο 5**, μέσα από κομμάτια κώδικα, και στιγμιότυπα (screenshots) από τη δομή των φακέλων παρουσιάστηκε η υλοποίηση της ανάπτυξης του κώδικα τόσο στο επίπεδο του Χρήστη (Angular), όσο και στο επίπεδο του Διακομιστή (Nest.JS). Επιπλέον παρουσιάστηκε η αξιολόγηση της εφαρμογής από ένα τυχαίο δείγμα είκοσι ανθρώπων, για τους οποίους αποδείχτηκε 50% πιθανότητα να προτείνουν την εφαρμογή σε γνωστούς τους και 80% ικανοποίηση και ευκολία χρήσης της.

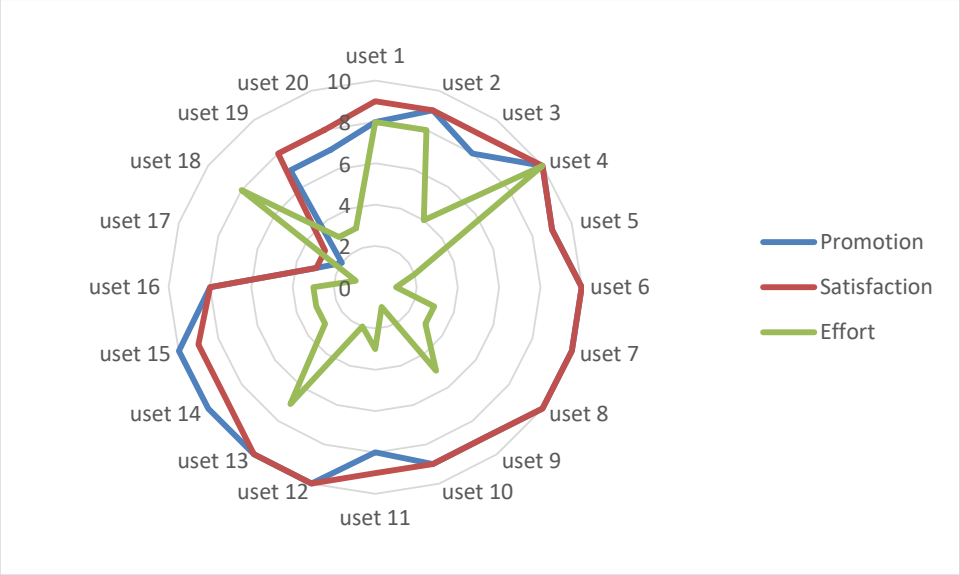
Στο **Κεφάλαιο 6** η θεωρία των παραπάνω κεφαλαίων πήρε υπόσταση, αναλύοντας και

παρουσιάζοντας την εφαρμογή όπως αναπτύχθηκε για τους σκοπούς της επίδειξης. Μέσα από στιγμιότυπα (screenshots) της εφαρμογής και χάρτες (site-maps) των σελίδων, παρατέθηκαν οι οδηγίες χρήσης της εφαρμογής, και δόθηκε μια πρώτη εντύπωση για την όψη της. Αναφέρθηκαν οι διάφορες βιβλιοθήκες που είναι δυνατόν να χρησιμοποιηθούν από την Angular, και οι λόγοι για τους οποίους καταλήξαμε να επιλέξουμε τη «Semantic UI». Επιπλέον, επισημάνθηκε η υποστήριξη δύο Διεπαφών (interfaces) από την εφαρμογή, αυτή του χρήστη/πελάτη και αυτή του διαχειριστή/διοργανωτή, και παρουσιάστηκε κάθε μία ξεχωριστά.

7.2 Αξιολόγηση

Για την λήψη ανατροφοδότησης από τους χρήστες της εφαρμογής, επιλέχθηκαν τρείς μετρικές σε ένα τυχαίο δείγμα είκοσι ανθρώπων (βλ. Παρ. Α), στους οποίους κοινοποιήθηκε ένα διαδραστικό «Google Form» προκειμένου να βαθμολογήσουν την εφαρμογή. Για την απεικόνιση των αποτελεσμάτων επιλέχθηκαν τα γραφήματα αράχνης (Εικόνα 5.12) και γραμμής (Εικόνα 5.13), όπου στον κατακόρυφο άξονα **y** φαίνεται η βαθμολογία των χρηστών και στον οριζόντιο άξονα **x** οι χρήστες. Οι μετρικές αυτές που είναι κοινά χρησιμοποιούμενες στην ανάπτυξη διαδικτυακών εφαρμογών, και έχουν κανονικοποιηθεί στην κλίμακα από μηδέν έως ένα (0 - 1) με καλύτερη την τιμή ένα, είναι οι παρακάτω:

- Βαθμολογία Διαδικτυακής Προώθησης **NPS** (Net Promoter Score), η οποία αφορά τη βαθμολόγηση της πιθανότητας του χρήστη να προτείνει την εφαρμογή σε άλλους. Υπολογίζεται από την αφαίρεση του ποσοστού των αρνητικών βαθμολογιών (0-6) από το ποσοστού των θετικών βαθμολογιών (9 ή 10). Σχετικά με την παρούσα εφαρμογή έχουμε $\mathbf{NPS = 0.9 - 0.1 = 0.8}$, που ισοδυναμεί με **80%** πιθανότητα οι χρήστες να προτείνουν την εφαρμογή.
- Βαθμολογία ικανοποίησης χρηστών **CSAT** (Customer Satisfaction Score), η οποία βαθμολογεί την ικανοποίηση των χρηστών από τη χρήση της εφαρμογής, και υπολογίζεται από το μέσο όρο της βαθμολογίας τους. Για την εφαρμογή μας, αυτή ισούται με $\boldsymbol{CSAT = 0,855}$ που ισοδυναμεί με **85%** ικανοποίησης των χρηστών.
- Βαθμολογία Καταβαλλόμενης Προσπάθειας **CES** (Customer Effort Score), μετρική η οποία βαθμολογεί την καταβαλλόμενη προσπάθεια των χρηστών για να χρησιμοποιήσουν την εφαρμογή, δηλώνοντας τη δυσκολία χρήσης της. Για την εφαρμογή της παρούσας εργασίας έχουμε $\boldsymbol{CES = 4,05}$.



Εικόνα 5. 12: Γράφημα Αράχνης για την αξιολόγηση της εφαρμογής.

Εικόνα 5. 13: Γραμμικό γράφημα για την αξιολόγηση της εφαρμογής.

Άλλες μετρικές που μπορεί να χρησιμοποιηθούν μετά το deploy της εφαρμογής στο Διαδίκτυο, σχετίζονται με την επαναχρησιμοποίηση της εφαρμογής, όπως είναι οι active users, session duration, bounce rate, pages per session, retention rate και churn rate και με την ανατροφοδότηση από το χρήστη γενικότερα, όπως surveys, reviews, ratings, comments, emails, chats, και social media [39]. Με βάση τις παραπάνω μετρικές, ως προς την αξιολόγηση της παρούσας εφαρμογής, παρατηρείται μια τάση προτίμησης της εφαρμογής με πιθανότητα να προταθεί σε γνωστούς 80%, ικανοποίησης από τη χρήση της εφαρμογής περίπου 85% και ευκολίας χρήσης της περίπου 60%, καθιστώντας την μια αξιόλογη λύση για την επίλυση του προβλήματος διαχείρισης κρατήσεων σε πολιτιστικές εκδηλώσεις.

7.3 Συμπεράσματα

Από την διαδικασία της ανάπτυξης και της αξιολόγησης εξάγονται κάποια συμπεράσματα, τα οποία

συμβάλλουν στα θέματα που προτείνονται στην Παράγραφο 7.4 για μελλοντική ανάπτυξη. Τα συμπεράσματα αυτά μπορούμε να τα κατηγοριοποιήσουμε σε πλεονεκτήματα και μειονεκτήματα της εφαρμογής.

Ως πλεονεκτήματα θεωρούνται τα παρακάτω:

- Διευκόλυνση των διαδικασιών αναζήτησης, κράτησης, παρακολούθησης των πολιτιστικών εκδηλώσεων από τη πλευρά του θεατή,
- Διευκόλυνση της οργάνωσης, ως προς την παρακολούθηση των εισιτηρίων, των θέσεων, των ημερομηνιών των πολιτιστικών εκδηλώσεων από την πλευρά του διοργανωτή,
- Διευκόλυνση χρηματικών συναλλαγών, και εξυπηρέτησης κοινού,
- Αμεσότητα στην αλληλεπίδραση του χρήστη,
- Με βάση την αξιολόγηση διαπιστώθηκε ευκολία χρήσης της εφαρμογής και υψηλή πιθανότητα να προταθεί σε φιλικά πρόσωπα

Ως μειονεκτήματα/ανησυχίες θεωρούνται τα παρακάτω:

- Ανάγκη για εξοικείωση με τη διεπαφή της εφαρμογής, ακόμα και για χρήστες μεγάλης ηλικίας, οι οποίοι σε γενικές γραμμές αντιμετωπίζουν δυσκολίες με τα σύγχρονα πληροφοριακά συστήματα,
- Η εφαρμογή θα πρέπει να ενημερώνεται τακτικά, ώστε να πληροί τις προδιαγραφές ασφαλείας, και να καλύπτει τυχόν κενά που δεν λήφθηκαν υπόψη κατά τη διάρκεια της ανάπτυξης,
- Από τα παραπάνω συμπεραίνουμε πως απαιτείται προσωπικό συντήρησης της εφαρμογής.

Τα συμπεράσματα αυτά είναι τα κυριότερα, και συνοψίζουν συνοπτικά την αποτελεσματικότητα της εφαρμογής και τη θετική επίδραση στην συμμετοχή των πολιτών σε πολιτιστικές εκδηλώσεις που λαμβάνουν χώρα σε εθνικό επίπεδο.

7.4 Μελλοντική Ανάπτυξη

Με σκοπό την εκπλήρωση των απαιτήσεων της σύγχρονης αγοράς για τη συμμετοχή σε πολιτιστικές εκδηλώσεις, η εργασία αυτή παρουσίασε το σχεδιασμό και την υλοποίηση μιας Διαδικτυακής εφαρμογής Διαχείρισης Κρατήσεων σε Πολιτιστικές Εκδηλώσεις, όπως είναι οι θεατρικές παραστάσεις. Επιπλέον, έθεσε τα θεμέλια για την αξιοποίηση μιας εφαρμογής Διαχείρισης Κρατήσεων Πολιτιστικών Εκδηλώσεων, με την αξιοποίηση της στοίβας ανάπτυξης MEAN και συγκεκριμένα των τεχνολογιών MongoDB, Express.JS/Nest.JS, Angular, και Node.JS. Στην πορεία της ανάπτυξης, ήρθαμε αντιμέτωποι με θέματα που χρήζουν περαιτέρω έρευνα και ανάπτυξη. Μερικά από τα παραπάνω είναι τα ακόλουθα:

- Στις περισσότερες εφαρμογές οι ρόλοι είναι περισσότεροι, καθώς η συντήρηση μπορεί να υλοποιείται από ένα μεγάλο οργανισμό με συγκεκριμένη ιεραρχία και δικαιώματα. Σε αυτό το πλαίσιο, περισσότερη ανάλυση των ρόλων των χρηστών στην εφαρμογή και περαιτέρω ανάπτυξη διεπαφών (UI), θα εξασφάλιζε την καλή λειτουργία της εφαρμογής και της αποφυγής ανθρώπινων λαθών, με δυσμενείς συνέπειες όπως η διαγραφή/καταστροφή δεδομένων από χρήστη με παραπάνω δυνατότητες από αυτές που θα όφειλε να έχει.

- Για τους χρήστες επίσης, η διαχείριση προφίλ είναι πολύ σημαντική, με δυνατότητες όπως η αλλαγή password, email, username, εικόνα προφίλ κ.α.
- Η MongoDB είναι ένα Σύστημα Διαχείρισης Βάσεων Δεδομένων (ΣΔΒΔ), με εξαιρετικές δυνατότητες επεξεργασίας και προσπέλασης μεγάλων δεδομένων (Big Data), όπως είναι τα οι αναφορές (logs) καλής λειτουργίας των υπηρεσιών και της ίδιας της εφαρμογής. Για τον λόγο αυτό, διακρίνεται επίσης στους τομείς της Στατιστικής και Ανάλυσης Δεδομένων, γεγονός που ανοίγει το δρόμο για μελλοντική αναβάθμιση και στροφή της εφαρμογής και στους παραπάνω τομείς
- Μια άλλη δυνατότητα που συναντάται σε εφαρμογές ηλεκτρονικού εμπορίου, είναι η ανάρτηση σχολίων από τους πελάτες για τα προϊόντα που παρατίθενται σε αυτές. Θα μπορούσε να εξεταστεί η δυνατότητα αυτή και για την εφαρμογή διαχείρισης κρατήσεων πολιτιστικών εκδηλώσεων, ωθώντας σε αγορά εισιτηρίων σε εκδηλώσεις με καλά σχόλια και αποτρέποντας σε άλλες με άσχημα σχόλια, αποτρέποντας με αυτόν τον τρόπο αρνητική κριτική και στην εφαρμογή.

Αναφορές

[1] "Introduction to the server side - Learn web development | MDN." https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction (accessed Aug. 27, 2023).

[2] "Three-tier architecture overview - AWS Serverless Multi-Tier Architectures with Amazon API Gateway and AWS Lambda." https://docs.aws.amazon.com/whitepapers/latest/serverless-multi-tier-architectures-api-gateway-lambda/three-tier-architecture-overview.html (accessed Aug. 17, 2023).

[3] "Single-page application - Wikipedia." https://en.wikipedia.org/wiki/Single-page_application (accessed Aug. 20, 2023).

[4] "What Is Big Data? | Oracle." https://www.oracle.com/big-data/what-is-big-data/ (accessed Aug. 20, 2023).

[5] Hazzlenut, "MongoDB: An Overview for Big Data Storage and Analytics | by Hazzlenut | Medium," 2023. https://hazzlenut.medium.com/mongodb-an-overview-for-big-data-storage-and-analytics-9142045a62d2 (accessed Aug. 20, 2023).

[6] JWT documentation, "JSON Web Token Introduction - jwt.io," 2021. https://jwt.io/introduction (accessed Oct. 05, 2021).

[7] Schmedtmann J, "Node.js, Express, MongoDB & More: The Complete Bootcamp 2021 | Udemy [MOOC]," 2021. https://www.udemy.com/course/nodejs-express-mongodb-bootcamp/ (accessed Oct. 05, 2021).

[8] "What is a REST API?" https://www.redhat.com/en/topics/api/what-is-a-rest-api (accessed Sep. 10, 2023).

[9] "MongoDB: The Developer Data Platform | MongoDB." https://www.mongodb.com/ (accessed Aug. 26, 2023).

[10] "Content management system - Wikipedia." https://en.wikipedia.org/wiki/Content_management_system (accessed Aug. 26, 2023).

[11] "Mongoose ODM v7.4.5." https://mongoosejs.com/ (accessed Aug. 26, 2023).

[12] Schmedtmann J, "Node.js, Express, MondoDB & More: The Complete Bootcamp[MOOC]," 2021. https://www.udemy.com/course/nodejs-express-mongodb-bootcamp/

[13] "Documentation | Node.js." https://nodejs.org/en/docs (accessed Aug. 27, 2023).

[14] "NestJS - A progressive Node.js framework." https://nestjs.com/ (accessed Aug. 26, 2023).

[15] "Google - Wikipedia." https://en.wikipedia.org/wiki/Google (accessed Aug. 26, 2023).

[16] "Angular - What is Angular?," 2021. https://angular.io/guide/what-is-angular (accessed Oct. 05, 2021).

[17] "Introduction to the DOM - Web APIs | MDN." https://developer.mozilla.org/en-US/docs/Web/API/Document_Object_Model/Introduction (accessed Aug. 26, 2023).

[18] "What is an API? - Application Programming Interfaces Explained - AWS." https://aws.amazon.com/what-is/api/ (accessed Aug. 26, 2023).

[19] "EventTarget: addEventListener() method - Web APIs | MDN." https://developer.mozilla.org/en-US/docs/Web/API/EventTarget/addEventListener (accessed Aug. 26, 2023).

[20] "RxJS - Observable." https://rxjs.dev/guide/observable (accessed Aug. 26, 2023).

[21] "Web design issues; What a semantic can represent." https://www.w3.org/DesignIssues/RDFnot.html (accessed Aug. 27, 2023).

[22] P. Hu and G. U. Dongxiao, "Development and Implementation of WEB-based Online Hotel Reservation System," pp. 235–238, Feb. 2013, doi: 10.2991/ISCCCA.2013.58.

[23] V. DeBolt, *Mastering Integrated HTML and CSS*. 2007. [Online]. Available: https://books.google.gr/books?id=U66yNAEACAAJ

[24] P. O. O. D. O. Kalu, Constance Tim, "Development Of Progressive Web Application For Tourism Reservation Management System," *Sci. Technol. Publ.*, vol. 7, no. 3, 2023.

[25] "What is .NET Framework? A software development framework." https://dotnet.microsoft.com/en-us/learn/dotnet/what-is-dotnet-framework (accessed Aug. 27, 2023).

[26] C. Feng, Zhou Takeshi, "A Web Application Framework for Reservation Systems and its Reusability Evaluation," in *Lecture Notes in Engineering and Computer Science*, 2009, p. 2174.

[27] “JavaServer Pages Technology - The Java EE 5 Tutorial.” https://docs.oracle.com/javaee/5/tutorial/doc/bnagx.html (accessed Aug. 27, 2023).
[28] S. Prajapati, A., Dhirani, K., Agrawal, M., Gurwara, N., & Tak, “A systematic review on online airline reservation system,” *Int. J. Adv. Res. Ideas Innov. Technol.*, vol. 4, pp. 915–918, 2018.
[29] S. Al-Fedaghi, “Developing Web Applications,” *Int. J. Softw. Eng. Its Appl.*, vol. 5, 2011, doi: 10.1007/978-1-4302-3531-6_12.
[30] P. Zhao, I. Yoo, J. Lavoie, B. J. Lavoie, and E. Simoes, “Web-Based Medical Appointment Systems: A Systematic Review,” *J Med Internet Res 2017;19(4)e134 https//www.jmir.org/2017/4/e134*, vol. 19, no. 4, p. e6747, Apr. 2017, doi: 10.2196/JMIR.6747.
[31] J. Muittari, “MODERN WEB BACK-END,” Oulu University of Applied Sciences, 2020.
[32] A. Adhikari, “Full Stack JavaScript: Web Application Development with MEAN,” Helsinki Metropolia University of Applied Sciences, 2016.
[33] N. Le Thanh, “MEAN STACK WEB DEVELOPMENT,” CENTRIA UNIVERSITY OF APPLIED SCIENCES, 2016.
[34] S. Naikwadi *et al.*, “COLLEGE MANAGEMENT WEB APPLICATION SYSTEM USING MEAN STACK,” *JETIR*, vol. 9, no. 5, pp. b476–b481, 2022, Accessed: Aug. 27, 2023. [Online]. Available: https://www.jetir.org/view?paper=JETIR2205165
[35] P. R. M. K. Manish Bagal, Gaurav Kadam, Ajay Bandgar, Abhijit Salunkhe, “A Survey Paper on Full Stack Web Development,” *Int. J. Sci. Res. Eng. Trends*, vol. 7, no. 3, 2021.
[36] M. P. S. Mr. Ninaad Nirgudkar, “The MEAN Stack,” *Int. Res. J. Eng. Technol.*, vol. 4, no. 5, 2017.
[37] “async function - JavaScript | MDN.” https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Statements/async_function (accessed Aug. 26, 2023).
[38] “HTTP response status codes - HTTP | MDN.” https://developer.mozilla.org/en-US/docs/Web/HTTP/Status (accessed Aug. 26, 2023).
[39] “What are the best metrics to test user satisfaction of your web app?” https://www.linkedin.com/advice/0/what-best-metrics-test-user-satisfaction-your (accessed Sep. 12, 2023).
[40] “CSS - Wikipedia.” https://en.wikipedia.org/wiki/CSS (accessed Aug. 20, 2023).
[41] “Internet Traffic from Mobile Devices (Sept 2023).” https://explodingtopics.com/blog/mobile-internet-traffic (accessed Sep. 23, 2023).

Παράρτημα Α. Ανατροφοδότηση Χρήσης της Εφαρμογής

Το Παράρτημα αυτό παρουσιάζει τον πίνακα με τη βαθμολόγηση είκοσι χρηστών της εφαρμογής, και αφορά την πιθανότητα να προτείνουν την εφαρμογή σε γνωστούς τους (Promotion), την ικανοποίηση τους από τη χρήση της (Satisfaction), και τη δυσκολία χρήσης της (effort).

Πίνακας Α.1

	Promotion	Satisfaction	Effort
user1	8	9	8
user2	9	9	8
user3	8	9	4
user4	10	10	10
user5	9	9	2
user6	10	10	1
user7	10	10	3
user8	10	10	3
user9	9	9	5
user10	9	9	1
user11	8	9	3
user12	10	10	2
user13	10	10	7
user14	10	9	3
user15	10	9	3
user16	8	8	3
user17	3	3	1
user18	2	3	8
user19	7	8	3
user20	7	8	3
Μέσος Όρος	8,35	8,55	4,05

Η απαντήσεις τους υποβλήθηκαν σε φόρμα google (Google Form) και προβλήθηκαν σε γραφήματα με τη χρήση του Microsoft Office (Excel).